NAACL HLT 2019

The International Workshop on Semantic Evaluation

Proceedings of the Thirteenth Workshop

June 6–June 7, 2019
Minneapolis, Minnesota, USA

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
acl@aclweb.org

ISBN 978-1-950737-06-2

Introduction

Welcome to SemEval-2019!

The Semantic Evaluation (SemEval) series of workshops focuses on the evaluation and comparison of systems that can analyse diverse semantic phenomena in text with the aim of extending the current state of the art in semantic analysis and creating high quality annotated datasets in a range of increasingly challenging problems in natural language semantics. SemEval provides an exciting forum for researchers to propose challenging research problems in semantics and to build systems/techniques to address such research problems.

SemEval-2019 is the thirteenth workshop in the series of International Workshops on Semantic Evaluation. The first three workshops, SensEval-1 (1998), SensEval-2 (2001), and SensEval-3 (2004), focused on word sense disambiguation, each time growing in the number of languages offered, in the number of tasks, and also in the number of participating teams. In 2007, the workshop was renamed to SemEval, and the subsequent SemEval workshops evolved to include semantic analysis tasks beyond word sense disambiguation. In 2012, SemEval turned into a yearly event. It currently runs every year, but on a two-year cycle, i.e., the tasks for SemEval 2019 were proposed in 2018.

SemEval-2019 was co-located with the 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2019) in Minneapolis, Minnesota, USA. It included the following 11 shared tasks organized in five tracks:

- Frame Semantics and Semantic Parsing
  - Task 1: Cross-lingual Semantic Parsing with UCCA
  - Task 2: Unsupervised Lexical Semantic Frame Induction
- Opinion, Emotion and Abusive Language Detection
  - Task 3: EmoContext: Contextual Emotion Detection in Text
  - Task 4: Hyperpartisan News Detection
  - Task 5: HatEval: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter
  - Task 6: OffensEval: Identifying and Categorizing Offensive Language in Social Media
- Fact vs. Fiction
  - Task 7: RumourEval 2019: Determining Rumour Veracity and Support for Rumours
  - Task 8: Fact Checking in Community Question Answering Forums
- Information Extraction and Question Answering
  - Task 9: Suggestion Mining from Online Reviews and Forums
  - Task 10: Math Question Answering
- NLP for Scientific Applications
  - Task 12: Toponym Resolution in Scientific Papers

This volume contains both Task Description papers that describe each of the above tasks, and System Description papers that present the systems that participated in these tasks. A total of 11 task description papers and 220 system description papers are included in this volume.

We are grateful to all task organizers as well as to the large number of participants whose enthusiastic participation has made SemEval once again a successful event. We are thankful to the task organizers who also served as area chairs, and to task organizers and participants who reviewed paper submissions. These proceedings have greatly benefited from their detailed and thoughtful feedback. We also thank the NAACL HLT 2019 conference organizers for their support. Finally, we most gratefully acknowledge the support of our sponsors: the ACL Special Interest Group on the Lexicon (SIGLEX) and Microsoft.

The SemEval 2019 organizers, Jonathan May, Ekaterina Shutova, Aurelie Herbelot, Xiaodan Zhu, Marianna Apidianaki, Saif M. Mohammad

Task Organizers:

Omri Abend, Hebrew University of Jerusalem, Israel
Payam Adineh, Bauhaus-Universität Weimar, Germany
Puneet Agrawal, Microsoft, India
Zohar Aizenbud, Hebrew University of Jerusalem, Israel
Ahmet Aker, University of Sheffield, UK
Pepa Atanasova, University of Copenhagen, Denmark
Ramy Baly, MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA
Valerio Basile, Università degli Studi di Torino, Italy
Kalina Bontcheva, University of Sheffield, UK
Cristina Bosco, Università degli Studi di Torino, Italy
Paul Buitelaar, National University of Ireland, Ireland
Marie Candito, Paris Diderot University - CNRS, France
Ankush Chatterjee, Microsoft, India
Leshem Choshen, Hebrew University of Jerusalem, Israel
David Corney, Factmata Ltd., UK
Tobias Daudert, National University of Ireland, Ireland
Leon Derczynski, IT University of Copenhagen, Denmark
Noura Farra, Columbia University, USA
Elisabetta Fersini, Universitat Politècnica de València, Spain
Graciela Gonzalez-Hernandez, University of Pennsylvania, USA
Genevieve Gorrell, University of Sheffield, UK
Hannaneh Hajishirzi, University of Washington, USA
Daniel Hershcovich, Hebrew University of Jerusalem, Israel
Mark Hopkins, Reed College, USA
Meghana Joshi, Microsoft, India
Laura Kallmeyer, HHUD, SFB991, Germany
Georgi Karadzhov, SiteGround Hosting EOOD, Bulgaria
Johannes Kiesel, Bauhaus-Universität Weimar, Germany
Elena Kochkina, University of Warwick, UK; Alan Turing Institute, UK
Rik Koncel-Kedziorski, University of Washington, USA
Ritesh Kumar, Bhim Rao Ambedkar University, India
Ronan Le Bras, Allen Institute for Artificial Intelligence, USA
Maria Liakata, University of Warwick, UK; Alan Turing Institute, UK
Arjun Magge, Arizona State University, USA
Shervin Malmasi, Amazon Research, USA
Maria Mestre, Factmata Ltd., UK
Tsvetomila Mihaylova, Instituto de Telecomunicações, Portugal
Mitra Mohtarami, MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA
Preslav Nakov, HBKU, Qatar
Kedhar Nath Narahari, Microsoft, India
Sapna Negi, Genesys Telecommunication Laboratories, Ireland
Debora Nozza, Universitat Politècnica de València, Spain
Karen O'Connor, University of Pennsylvania, USA
Viviana Patti, Università degli Studi di Torino, Italy
Cristian Petrescu-Prahova, Allen Institute for Artificial Intelligence, USA
Miriam R. L. Petruck, ICSI, US
Martin Potthast, Leipzig University, Germany
Behrang QasemiZadeh, SFB991, Germany
Francisco Rangel, Università degli Studi di Milano Bicocca, Italy; Autoritas Consulting, Spain
Ari Rappoport, Hebrew University of Jerusalem, Israel

Sara Rosenthal, IBM Research, USA
Paolo Rosso, Università degli Studi di Milano Bicocca, Italy
Manuela Sanguinetti, Università degli Studi di Torino, Italy
Matthew Scotch, Arizona State University, USA
Rishabh Shukla, Factmata Ltd., UK
Gabriel Stanovsky, University of Washington, USA
Benno Stein, Bauhaus-Universität Weimar, Germany
Regina Stodden, HHUD, Germany
Elior Sulem, Hebrew University of Jerusalem, Israel
Emmanuel Vincent, Factmata Ltd., UK
Davy Weissenbacher, University of Pennsylvania, USA
Marcos Zampieri, University of Wolverhampton, UK
Arkaitz Zubiaga, Queen Mary University of London, UK

Invited Speaker:

Samuel R. Bowman, New York University

Invited Talk: Task-Independent Sentence Understanding

Samuel R. Bowman
New York University

Abstract

This talk deals with the goal of task-independent language understanding: building machine learning models that can learn to do most of the hard work of language understanding before they see a single example of the language understanding task they're meant to solve, in service of making the best of modern NLP systems both better and more data-efficient. I'll survey the (dramatic!) progress that the NLP research community has made toward this goal in the last year. In particular, I'll dwell on GLUE—an open-ended shared task competition that measures progress toward this goal for sentence understanding tasks—and I'll preview a few recent and forthcoming analysis papers that attempt to offer a bit of perspective on this recent progress.

Biography

I have been on the faculty at NYU since 2016, when I finished my PhD with Chris Manning and Chris Potts at Stanford. At NYU, I'm a core member of the new school-level Data Science unit, which focuses on machine learning, and a co-PI of the CILVR machine learning lab. My research focuses on data, evaluation techniques, and modeling techniques for sentence understanding in natural language processing, and on applications of machine learning to scientific questions in linguistic syntax and semantics. I am an area chair for *SEM 2018, ICLR 2019, and NAACL 2019; I organized a twenty-three person team at JSALT 2018; and I earned a 2015 EMNLP Best Resource Paper Award and a 2017 Google Faculty Research Award.

Table of Contents

Workshop Program

Thursday, June 6, 2019

09:00–09:15 ***Welcome / Opening Remarks***

09:30–10:30 *Invited Talk: Task-Independent Sentence Understanding*
Sam Bowman

10:30–11:00 ***Coffee***

11:00–12:30 ***Tasks 1, 2 and 3***

SemEval-2019 Task 1: Cross-lingual Semantic Parsing with UCCA
Daniel Hershcovich, Zohar Aizenbud, Leshem Choshen, Elior Sulem, Ari Rappoport and Omri Abend

HLT@SUDA at SemEval-2019 Task 1: UCCA Graph Parsing as Constituent Tree Parsing
Wei Jiang, Zhenghua Li, Yu Zhang and Min Zhang

SemEval-2019 Task 2: Unsupervised Lexical Frame Induction
Behrang QasemiZadeh, Miriam R L Petruck, Regina Stodden, Laura Kallmeyer and Marie Candito

Neural GRANNy at SemEval-2019 Task 2: A combined approach for better modeling of semantic relationships in semantic frame induction
Nikolay Arefyev, Boris Sheludko, Adis Davletov, Dmitry Kharchev, Alex Nevidomsky and Alexander Panchenko

SemEval-2019 Task 3: EmoContext Contextual Emotion Detection in Text
Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi and Puneet Agrawal

ANA at SemEval-2019 Task 3: Contextual Emotion detection in Conversations through hierarchical LSTMs and BERT
Chenyang Huang, Amine Trabelsi and Osmar Zaiane

12:30–14:00 ***Lunch***

14:00–15:30 ***Tasks 5 and 6***

Thursday, June 6, 2019 (continued)

Thursday, June 6, 2019 (continued)

Thursday, June 6, 2019 (continued)

Thursday, June 6, 2019 (continued)

Thursday, June 6, 2019 (continued)

Thursday, June 6, 2019 (continued)

Friday, June 7, 2019 (continued)

SemEval-2019 Task 1: Cross-lingual Semantic Parsing with UCCA

Daniel Hershcovich **Zohar Aizenbud** **Leshem Choshen**
Elior Sulem **Ari Rappoport** **Omri Abend**
School of Computer Science and Engineering
Hebrew University of Jerusalem
{danielh,zohara,borgr,eliors,arir,oabend}@cs.huji.ac.il

Abstract

We present the SemEval 2019 shared task on Universal Conceptual Cognitive Annotation (UCCA) parsing in English, German and French, and discuss the participating systems and results. UCCA is a cross-linguistically applicable framework for semantic representation, which builds on extensive typological work and supports rapid annotation. UCCA poses a challenge for existing parsing techniques, as it exhibits reentrancy (resulting in DAG structures), discontinuous structures and non-terminal nodes corresponding to complex semantic units. The shared task has yielded improvements over the state-of-the-art baseline in all languages and settings. Full results can be found in the task's website https://competitions.codalab.org/competitions/19160.

1 Overview

Semantic representation is receiving growing attention in NLP in the past few years, and many proposals for semantic schemes have recently been put forth. Examples include Abstract Meaning Representation (AMR; Banarescu et al., 2013), Broad-coverage Semantic Dependencies (SDP; Oepen et al., 2016), Universal Decompositional Semantics (UDS; White et al., 2016), Parallel Meaning Bank (Abzianidze et al., 2017), and Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport, 2013). These advances in semantic representation, along with corresponding advances in semantic parsing, can potentially benefit essentially all text understanding tasks, and have already demonstrated applicability to a variety of tasks, including summarization (Liu et al., 2015; Dohare and Karnick, 2017), paraphrase detection (Issa et al., 2018), and semantic evaluation (using UCCA; see below). In this shared task, we focus on UCCA parsing in multiple languages.

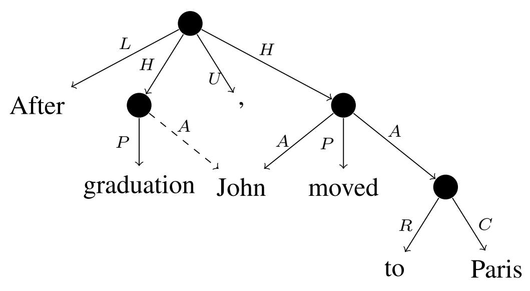


Figure 1: An example UCCA graph.

One of our goals is to benefit semantic parsing in languages with less annotated resources by making use of data from more resource-rich languages. We refer to this approach as *cross-lingual* parsing, while other works (Zhang et al., 2017, 2018) define cross-lingual parsing as the task of parsing text in one language to meaning representation in another language.

In addition to its potential applicative value, work on semantic parsing poses interesting algorithmic and modeling challenges, which are often different from those tackled in syntactic parsing, including reentrancy (e.g., for sharing arguments across predicates), and the modeling of the interface with lexical semantics.

UCCA is a cross-linguistically applicable semantic representation scheme, building on the established Basic Linguistic Theory typological framework (Dixon, 2010b,a, 2012). It has demonstrated applicability to multiple languages, including English, French and German, and pilot annotation projects were conducted on a few languages more. UCCA structures have been shown to be well-preserved in translation (Sulem et al., 2015), and to support rapid annotation by non-experts, assisted by an accessible annotation interface (Abend et al., 2017).[1] UCCA has already shown applicative value for text simplifica-

[1] https://github.com/omriabnd/UCCA-App

		Scene Elements
P	**Process**	The main relation of a Scene that evolves in time (usually an action or movement).
S	**State**	The main relation of a Scene that does not evolve in time.
A	**Participant**	Scene participant (including locations, abstract entities and Scenes serving as arguments).
D	**Adverbial**	A secondary relation in a Scene.
		Elements of Non-Scene Units
C	**Center**	Necessary for the conceptualization of the parent unit.
E	**Elaborator**	A non-Scene relation applying to a single Center.
N	**Connector**	A non-Scene relation applying to two or more Centers, highlighting a common feature.
R	**Relator**	All other types of non-Scene relations: (1) Rs that relate a C to some super-ordinate relation, and (2) Rs that relate two Cs pertaining to different aspects of the parent unit.
		Inter-Scene Relations
H	**Parallel Scene**	A Scene linked to other Scenes by regular linkage (e.g., temporal, logical, purposive).
L	**Linker**	A relation between two or more Hs (e.g., "when", "if", "in order to").
G	**Ground**	A relation between the speech event and the uttered Scene (e.g., "surprisingly").
		Other
F	**Function**	Does not introduce a relation or participant. Required by some structural pattern.

Table 1: The complete set of categories in UCCA's foundational layer.

tion (Sulem et al., 2018b), as well as for defining semantic evaluation measures for text-to-text generation tasks, including machine translation (Birch et al., 2016), text simplification (Sulem et al., 2018a) and grammatical error correction (Choshen and Abend, 2018).

The shared task defines a number of tracks, based on the different corpora and the availability of external resources (see §5). It received submissions from eight research groups around the world. In all settings at least one of the submitted systems improved over the state-of-the-art TUPA parser (Hershcovich et al., 2017, 2018), used as a baseline.

2 Task Definition

UCCA represents the semantics of linguistic utterances as directed acyclic graphs (DAGs), where terminal (childless) nodes correspond to the text tokens, and non-terminal nodes to semantic units that participate in some super-ordinate relation. Edges are labeled, indicating the role of a child in the relation the parent represents. Nodes and edges belong to one of several *layers*, each corresponding to a "module" of semantic distinctions.

UCCA's *foundational layer* covers the predicate-argument structure evoked by predicates of all grammatical categories (verbal, nominal, adjectival and others), the inter-relations between them, and other major linguistic phenomena such as semantic heads and multi-word expressions. It is the only layer for which annotated corpora exist at the moment, and is thus the target of this shared task. The layer's basic notion is the *Scene*, describing a state, action, movement or some other relation that evolves in time. Each Scene contains one main relation (marked as either a Process or a State), as well as one or more Participants. For example, the sentence "After graduation, John moved to Paris" (Figure 1) contains two Scenes, whose main relations are "graduation" and "moved". "John" is a Participant in both Scenes, while "Paris" only in the latter. Further categories account for inter-Scene relations and the internal structure of complex arguments and relations (e.g., coordination and multi-word expressions). Table 1 provides a concise description of the categories used by the UCCA foundational layer.

UCCA distinguishes *primary* edges, corresponding to explicit relations, from *remote* edges (appear dashed in Figure 1) that allow for a unit to participate in several super-ordinate relations. Primary edges form a tree in each layer, whereas remote edges enable reentrancy, forming a DAG.

UCCA graphs may contain *implicit* units with no correspondent in the text. Figure 2 shows the annotation for the sentence "A similar technique is almost impossible to apply to other crops, such as cotton, soybeans and rice."[2] It includes a single Scene, whose main relation is "apply", a secondary relation "almost impossible", as well as two complex arguments: "a similar technique" and the coordinated argument "such as cotton, soybeans, and rice." In addition, the Scene includes an implicit argument, which represents the agent of the "apply" relation.

While parsing technology is well-established

[2]The same example was used by Oepen et al. (2015) to compare different semantic dependency schemes.

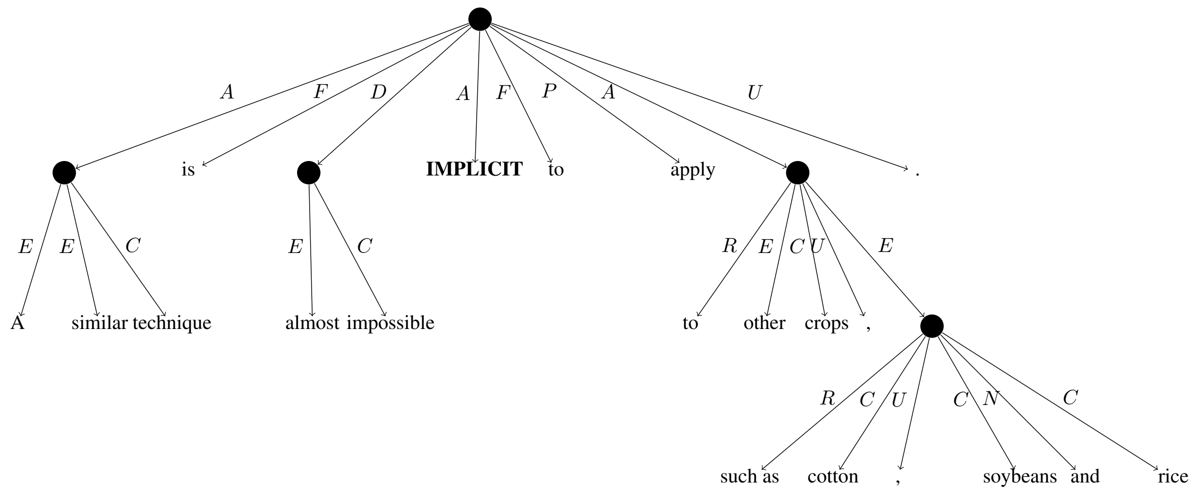


Figure 2: UCCA example with an implicit unit.

for syntactic parsing, UCCA has several formal properties that distinguish it from syntactic representations, mostly UCCA's tendency to abstract away from syntactic detail that do not affect argument structure. For instance, consider the following examples where the concept of a Scene has a different rationale from the syntactic concept of a clause. First, non-verbal predicates in UCCA are represented like verbal ones, such as when they appear in copula clauses or noun phrases. Indeed, in Figure 1, "graduation" and "moved" are considered separate Scenes, despite appearing in the same clause. Second, in the same example, "John" is marked as a (remote) Participant in the graduation Scene, despite not being explicitly mentioned. Third, consider the possessive construction in "John's trip home". While in UCCA "trip" evokes a Scene in which "John" is a Participant, a syntactic scheme would analyze this phrase similarly to "John's shoes".

The differences in the challenges posed by syntactic parsing and UCCA parsing, and more generally by semantic parsing, motivate the development of targeted parsing technology to tackle it.

3 Data & Resources

All UCCA corpora are freely available.[3] For English, we use v1.2.3 of the Wikipedia UCCA corpus (*Wiki*), v1.2.2 of the UCCA *Twenty Thousand Leagues Under the Sea* English-French parallel corpus (*20K*), which includes UCCA manual annotation for the first five chapters in French and English, and v1.0.1 of the UCCA German *Twenty Thousand Leagues Under the Sea* corpus, which includes the entire book in German. For consistent annotation, we replace any Time and Quantifier labels with Adverbial and Elaborator in these data sets. The resulting training, development[4] and test sets[5] are publicly available, and the splits are given in Table 2. Statistics on various structural properties are given in Table 3.

The corpora were manually annotated according to v1.2 of the UCCA guidelines,[6] and reviewed by a second annotator. All data was passed through automatic validation and normalization scripts.[7] The goal of validation is to rule out cases that are inconsistent with the UCCA annotation guidelines. For example, a Scene, defined by the presence of a Process or a State, should include at least one Participant.

Due to the small amount of annotated data available for French, we only provided a minimal training set of 15 sentences, in addition to the development and test set. Systems for French were expected to pursue semi-supervised approaches, such as cross-lingual learning or structure projection, leveraging the parallel nature of the corpus, or to rely on datasets for related formalisms, such as Universal Dependencies (Nivre et al., 2016). The full unannotated 20K Leagues corpus in English and French was released as well, in order to facilitate pursuing cross-lingual approaches.

Datasets were released in an XML format, including tokenized text automatically pre-

[3]`https://github.com/UniversalConceptualCognitiveAnnotation`

[4]`http://bit.ly/semeval2019task1traindev`

[5]`http://bit.ly/semeval2019task1test`

[6]`http://bit.ly/semeval2019task1guidelines`

[7]`https://github.com/huji-nlp/ucca/tree/master/scripts`

	train/trial		dev		test		total		
corpus	sentences	tokens	sentences	tokens	sentences	tokens	passages	sentences	tokens
English-Wiki	4,113	124,935	514	17,784	515	15,854	367	5,142	158,573
English-20K	0	0	0	0	492	12,574	154	492	12,574
French-20K	15	618	238	6,374	239	5,962	154	492	12,954
German-20K	5,211	119,872	651	12,334	652	12,325	367	6,514	144,531

Table 2: Data splits of the corpora used for the shared task.

	En-Wiki	En-20K	Fr-20K	De-20K
# passages	367	154	154	367
# sentences	5,141	492	492	6,514
# tokens	158,739	12,638	13,021	144,529
# non-terminals	62,002	4,699	5,110	51,934
% discontinuous	1.71	3.19	4.64	8.87
% reentrant	1.84	0.89	0.65	0.31
# edges	208,937	16,803	17,520	187,533
% primary	97.40	96.79	97.02	97.32
% remote	2.60	3.21	2.98	2.68
by category				
% Participant	17.17	18.1	17.08	19.86
% Center	18.74	16.31	18.03	14.32
% Adverbial	3.65	5.25	4.18	5.67
% Elaborator	18.98	18.06	18.65	14.88
% Function	3.38	3.58	2.58	2.98
% Ground	0.03	0.56	0.37	0.57
% Parallel Scene	6.02	6.3	6.15	7.54
% Linker	2.19	2.66	2.57	2.49
% Connector	1.26	0.93	0.84	0.65
% Process	7.1	7.51	6.91	7.03
% Relator	8.58	8.09	9.6	7.54
% State	1.62	2.1	1.88	3.34
% Punctuation	11.28	10.55	11.16	13.15

Table 3: Statistics of the corpora used for the shared task.

processed using spaCy (see §5), and gold-standard UCCA annotation for the train and development sets.[8] To facilitate the use of existing NLP tools, we also released the data in SDP, AMR, CoNLL-U and plain text formats.

4 TUPA: The Baseline Parser

We use the TUPA parser, the only parser for UCCA at the time the task was announced, as a baseline (Hershcovich et al., 2017, 2018). TUPA is a transition-based DAG parser based on a BiLSTM-based classifier.[9] TUPA in itself has been found superior to a number of conversion-based parsers that use existing parsers for other formalisms to parse UCCA by constructing a two-way conversion protocol between the formalisms. It can thus be regarded as a strong baseline for system submissions to the shared task.

5 Evaluation

Tracks. Participants in the task were evaluated in four settings:

1. English in-domain setting, using the Wiki corpus.
2. English out-of-domain setting, using the Wiki corpus as training and development data, and 20K Leagues as test data.
3. German in-domain setting, using the 20K Leagues corpus.
4. French setting with no training data, using the 20K Leagues as development and test data.

In order to allow both even ground comparison between systems and using hitherto untried resources, we held both an **open** and a **closed** track for submissions in the English and German settings. Closed track submissions were only allowed to use the gold-standard UCCA annotation distributed for the task in the target language, and were limited in their use of additional resources. Concretely, the only additional data they were allowed to use is that used by TUPA, which consists of automatic annotations provided by spaCy:[10] POS tags, syntactic dependency relations, and named entity types and spans. In addition, the closed track only allowed the use of word embeddings provided by fastText (Bojanowski et al., 2017)[11] for all languages.

Systems in the open track, on the other hand, were allowed to use any additional resource, such as UCCA annotation in other languages, dictionaries or datasets for other tasks, provided that they make sure not to use any additional gold standard annotation over the same text used in the UCCA

[8] https://github.com/UniversalConceptualCognitiveAnnotation/docs/blob/master/FORMAT.md

[9] https://github.com/huji-nlp/tupa

[10] http://spacy.io

[11] http://fasttext.cc

corpora.[12] In both tracks, we required that submitted systems are not trained on the development data. We only held an open track for French, due to the paucity of training data. The four settings and two tracks result in a total of 7 competitions.

Scoring. The following scores an output graph $G_1 = (V_1, E_1)$ against a gold one, $G_2 = (V_2, E_2)$, over the same sequence of terminals (tokens) W. For a node v in V_1 or V_2, define $yield(v) \subseteq W$ as is its set of terminal descendants. A pair of edges $(v_1, u_1) \in E_1$ and $(v_2, u_2) \in E_2$ with labels (categories) ℓ_1, ℓ_2 is *matching* if $yield(u_1) = yield(u_2)$ and $\ell_1 = \ell_2$. Labeled precision and recall are defined by dividing the number of matching edges in G_1 and G_2 by $|E_1|$ and $|E_2|$, respectively. F_1 is their harmonic mean:

$$2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Unlabeled precision, recall and F_1 are the same, but without requiring that $\ell_1 = \ell_2$ for the edges to match. We evaluate these measures for primary and remote edges separately. For a more fine-grained evaluation, we additionally report precision, recall and F_1 on edges of each category.[13]

6 Participating Systems

We received a total of eight submissions to the different tracks: *MaskParse@Deskiñ* (Marzinotto et al., 2019) from Orange Labs and Aix-Marseille University, *HLT@SUDA* (Jiang et al., 2019) from Soochow University, *TüPa* (Pütz and Glocker, 2019) from the University of Tübingen, *UC Davis* (Yu and Sagae, 2019) from the University of California, Davis , *GCN-Sem* (Taslimipoor et al., 2019) from the University of Wolverhampton, *CUNY-PekingU* (Lyu et al., 2019) from the City University of New York and Peking University, *DANGNT@UIT.VNU-HCM* (Nguyen and Tran, 2019) from the University of Information Technology VNU-HCM, and *XLangMo* from Zhejiang University. Some of the teams participated in more than one track and two systems (*HLT@SUDA* and *CUNY-PekingU*) participated in all the tracks.

In terms of parsing approaches, the task was quite varied. *HLT@SUDA* converted UCCA graphs to constituency trees and trained a constituency parser and a recovery mechanism of remote edges in a multi-task framework. *MaskParse@Deskiñ* used a bidirectional GRU tagger with a masking mechanism. *Tüpa* and *XLangMo* used a transition-based approach. *UC Davis* used an encoder-decoder architecture. *GCN-SEM* uses a BiLSTM model to predict Semantic Dependency Parsing tags, when the syntactic dependency tree is given in the input. *CUNY-PKU* is based on an ensemble that includes different variations of the TUPA parser. *DANGNT@UIT.VNU-HCM* converted syntactic dependency trees to UCCA graphs.

Different systems handled remote edges differently. *DANGNT@UIT.VNU-HCM* and *GCN-SEM* ignored remote edges. *UC Davis* used a different BiLSTM for remote edges. *HLT@SUDA* marked remote edges when converting the graph to a constituency tree and trained a classification model for their recovery. *MaskParse@Deskiñ* handles remote edges by detecting arguments that are outside of the parent's node span using a detection threshold on the output probabilities.

In terms of using the data, all teams but one used the UCCA XML format, two used the CoNLL-U format, which is derived by a lossy conversion process, and only one team found the other data formats helpful. One of the teams (*MaskParse@Deskiñ*) built a new training data adapted to their model by repeating each sentence N times, N being the number of non-terminal nodes in the UCCA graphs. Three of the teams adapted the baseline TUPA parser, or parts of it to form their parser, namely *TüPa*, *CUNY-PekingU* and *XLangMo*; *HLT@SUDA* used a constituency parser (Stern et al., 2017) as a component in their model; *DANGNT@UIT.VNU-HCM* is a rule-based system over the Stanford Parser, and the rest are newly constructed parsers.

All teams found it useful to use external resources beyond those provided by the Shared Task. Four submissions used external embeddings, MUSE (Conneau et al., 2017) in the case of *MaskParse@Deskiñ* and *XLangMo*, ELMo (Peters et al., 2018) in the case of *TüPa*,[14] and BERT (Devlin et al., 2018) in the case of *HLT@SUDA*.

[12]We are not aware of any such annotation, but include this restriction for completeness.

[13]The official evaluation script providing both coarse-grained and fine-grained scores can be found in `https://github.com/huji-nlp/ucca/blob/master/scripts/evaluate_standard.py`.

[14]GCN-Sem used ELMo in the closed tracks, training on the available data.

		Labeled			Unlabeled		
#	Team	All	Prim.	Rem.	All	Prim.	Rem.
English-Wiki (closed)							
1	HLT@SUDA	77.4	77.9	52.2	87.2	87.9	52.5
2	baseline	72.8	73.3	47.2	85.0	85.8	48.4
3	Davis	72.2	73.0	0	85.5	86.4	0
4	CUNY-PekingU	71.8	72.3	49.5	84.5	85.2	50.1
5	DANGNT@UIT. VNU-HCM	70.0	70.7	0	81.7	82.6	0
6	GCN-Sem	65.7	66.4	0	80.9	81.8	0
English-Wiki (open)							
1	HLT@SUDA	80.5	81.0	58.8	89.7	90.3	60.7
2	CUNY-PekingU	80.0	80.2	66.6	89.4	89.9	67.4
3	baseline	73.5	73.9	53.5	85.1	85.7	54.3
3	TüPa	73.5	74.1	42.5	85.3	86.2	43.1
4	XLangMo	73.1	73.5	53.2	85.1	85.7	53.5
5	DANGNT@UIT. VNU-HCM	70.3	71.1	0	81.7	82.6	0
English-20K (closed)							
1	HLT@SUDA	72.7	73.6	31.2	85.2	86.4	32.1
2	baseline	67.2	68.2	23.7	82.2	83.5	24.3
3	CUNY-PekingU	66.9	67.9	27.9	82.3	83.6	29.0
4	GCN-Sem	62.6	63.7	0	80.0	81.4	0
English-20K (open)							
1	HLT@SUDA	76.7	77.7	39.2	88.0	89.2	41.4
2	CUNY-PekingU	73.9	74.6	45.7	86.4	87.4	48.1
3	TüPa	70.9	71.9	29.6	84.4	85.7	30.7
4	XLangMo	69.5	70.4	36.6	83.5	84.6	38.5
5	baseline	68.4	69.4	25.9	82.5	83.9	26.2
German-20K (closed)							
1	HLT@SUDA	83.2	83.8	59.2	92.0	92.6	60.9
2	CUNY-PekingU	79.7	80.2	59.3	90.2	90.9	59.9
3	baseline	73.1	73.6	47.8	85.9	86.7	48.2
4	GCN-Sem	71.0	72.0	0	85.1	86.2	0
German-20K (open)							
1	HLT@SUDA	84.9	85.4	64.1	92.8	93.4	64.7
2	CUNY-PekingU	84.1	84.5	66.0	92.3	93.0	66.6
3	baseline	79.1	79.6	59.9	90.3	91.0	60.5
4	TüPa	78.1	78.8	40.8	89.4	90.3	41.2
5	XLangMo	78.0	78.4	61.1	89.4	90.1	61.4
French-20K (open)							
1	CUNY-PekingU	79.6	80.0	64.5	89.1	89.6	71.1
2	HLT@SUDA	75.2	76.0	43.3	86.0	87.0	45.1
3	XLangMo	65.6	66.6	13.3	81.5	82.8	14.1
4	MaskParse@Deskiñ	65.4	66.6	24.3	80.9	82.5	25.8
5	baseline	48.7	49.6	2.4	74.0	75.3	3.2
6	TüPa	45.6	46.4	0	73.4	74.6	0

Table 4: Official F1-scores for each system in each track. Prim.: primary edges, Rem.: remote edges.

Other resources included additional unlabeled data (*TüPa* and *CUNY-PekingU*), a list of multi-word expressions (*MaskParse@Deskiñ*), and the Stanford parser in the case of *DANGNT@UIT.VNU-HCM*. Only *CUNY-PKU* used the *20K* unlabeled parallel data in English and French.

A common trend for many of the systems was the use of cross-lingual projection or transfer (*MaskParse@Deskiñ*, *HLT@SUDA*, *TüPa*, *GCN-Sem*, *CUNY-PKU* and *XLangMo*). This was necessary for French, and was found helpful for German as well (*CUNY-PKU*).

7 Results

Table 4 shows the labeled and unlabeled F1 for primary and remote edges, for each system in each track. Overall F1 (All) is the F1 calculated over both primary and remote edges. Full results are available online.[15]

Figure 3 shows the fine-grained evaluation by labeled F1 per UCCA category, for each system in each track. While Ground edges were uniformly difficult to parse due to their sparsity in the training data, Relators were the easiest for all systems, as they are both common and predictable. The Process/State distinction proved challenging, and most main relations were identified as the more common Process category. The winning system in most tracks (HLT@SUDA) performed better on almost all categories. Its largest advantage was on Parallel Scenes and Linkers, showing was especially successful at identifying Scene boundaries relative to the other systems, which requires a good understanding of syntax.

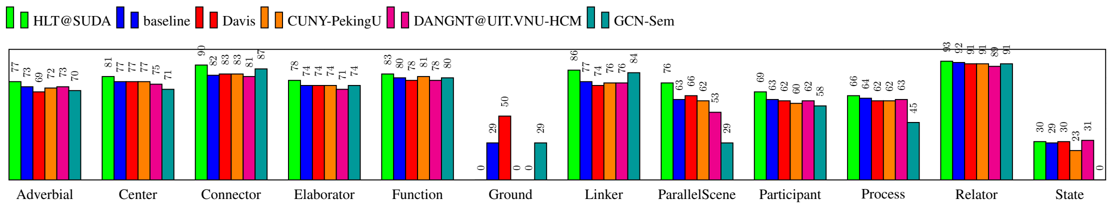
8 Discussion

The *HLT@SUDA* system participated in all the tracks, obtaining the first place in the six English and German tracks and the second place in the French open track. The system is based on the conversion of UCCA graphs into constituency trees, marking remote and discontinuous edges for recovery. The classification-based recovery of the remote edges is performed simultaneously with the constituency parsing in a multi-task learning framework. This work, which further connects between semantic and syntactic parsing, proposes a recovery mechanism that can be applied to other grammatical formalisms, enabling the conversion of a given formalism to another one for parsing. The idea of this system is inspired by the pseudo non-projective dependency parsing approach proposed by Nivre and Nilsson (2005).

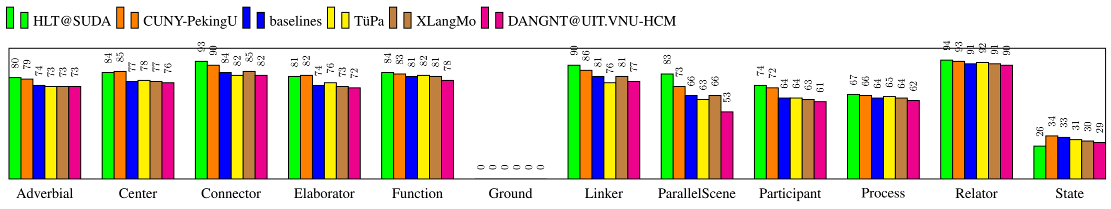
The *MaskParse@Deskiñ* system only participated to the French open track, focusing on cross-lingual parsing. The system uses a semantic tagger, implemented with a bidirectional GRU and a masking mechanism to recursively extract the inner semantic structures in the graph. Multilingual word embeddings are also used. Using the English and German training data as well as the small French trial data for training, the parser ranked fourth in the French open track with a labeled F1 score of 65.4%, suggesting that this new model could be useful for low-resource languages.

The *Tüpa* system takes a transition-based approach, building on the TUPA transition system and oracle, but modifies its feature representations. Specifically, instead of representing the parser configuration using LSTMs over the partially parsed graph, stack and buffer, they use feed-

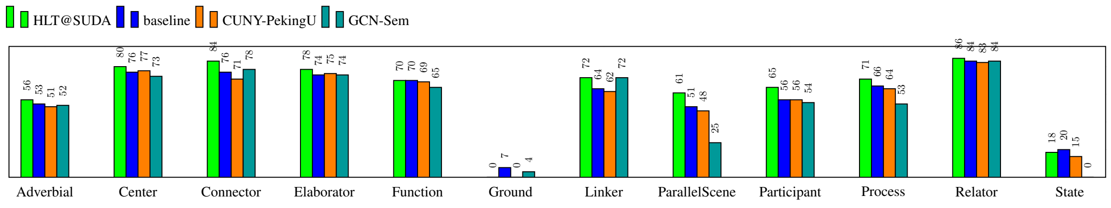
[15]http://bit.ly/semeval2019task1results



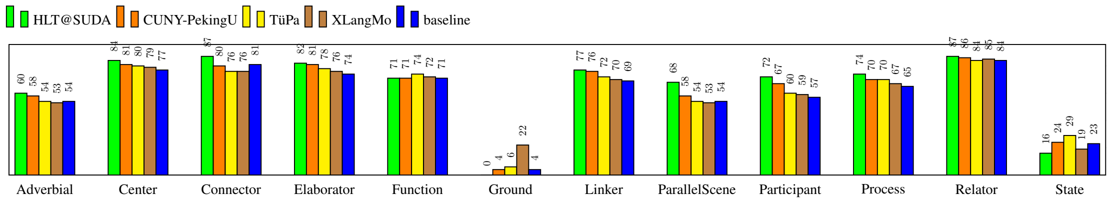
(a) English Wiki (closed)



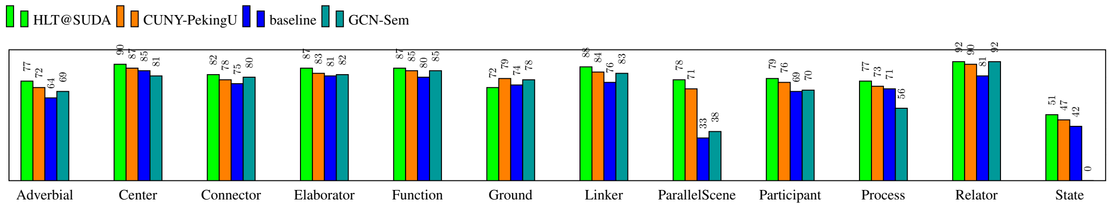
(b) English Wiki (open)



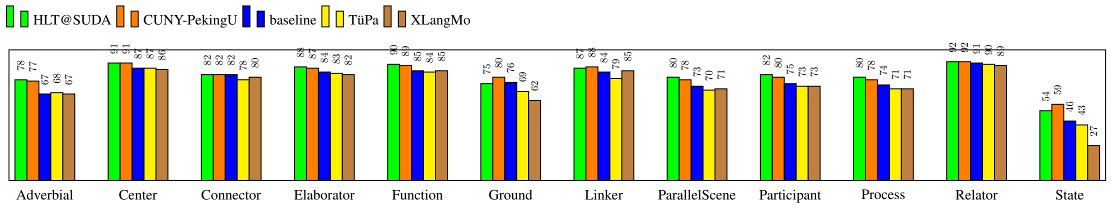
(c) English 20K (closed)



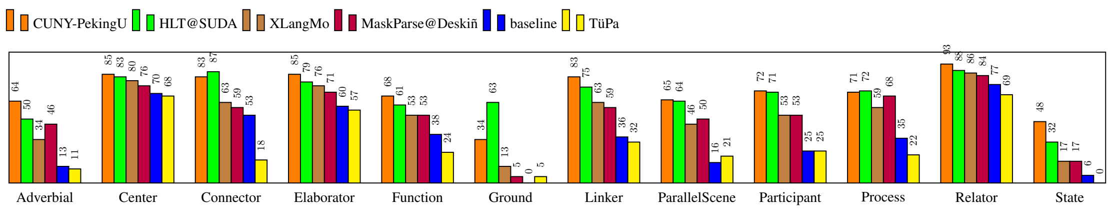
(d) English 20K (open)



(e) German 20K (closed)



(f) German 20K (open)



(g) French 20K (open)

Figure 3: Each system's labeled F1 per UCCA category in each track.

forward networks with ELMo contextualized embeddings. The stack and buffer are represented by the top three items on them. For the partially parsed graph, they extract the rightmost and leftmost parents and children of the respective items, and represent them by the ELMo embedding of their form, the embedding of their dependency heads (for terminals, for non-terminals this is replaced with a learned embedding) and the embeddings of all terminal children. Results are generally on-par with the TUPA baseline, and surpass it from the out-of-domain English setting. This suggests that the TUPA architecture may be simplified, without compromising performance.

The *UC Davis* system participated only in the English closed track, where they achieved the second highest score, on par with TUPA. The proposed parser has an encoder-decoder architecture, where the encoder is a simple BiLSTM encoder for each span of words. The decoder iteratively and greedily traverses the sentence, and attempts to predict span boundaries. The basic algorithm yields an unlabeled contiguous phrase-based tree, but additional modules predict the labels of the spans, discontiguous units (by joining together spans from the contiguous tree under a new node), and remote edges. The work is inspired by Kitaev and Klein (2018), who used similar methods for constituency parsing.

The *GCN-SEM* system uses a BiLSTM encoder, and predicts bi-lexical semantic dependencies (in the SDP format) using word, token and syntactic dependency parses. The latter is incorporated into the network with a graph convolutional network (GCN). The team participated in the English and German closed tracks, and were not among the highest-ranking teams. However, scores on the UCCA test sets converted to the bi-lexical CoNLL-U format were rather high, implying that the lossy conversion could be much of the reason.

The *CUNY-PKU* system was based on an ensemble. The ensemble included variations of TUPA parser, namely the MLP and BiLSTM models (Hershcovich et al., 2017) and the BiLSTM model with an additional MLP. The system also proposes a way to aggregate the ensemble going through CKY parsing and accounting for remotes and discontinuous spans. The team participated in all tracks, including additional information in the open domain, notably synthetic data based on automatically translating annotated texts. Their system ranks first in the French open track.

The *DANGNT@UIT.VNU-HCM* system participated only in the English Wiki open and closed tracks. The system is based on graph transformations from dependency trees into UCCA, using heuristics to create non-terminal nodes and map the dependency relations to UCCA categories. The manual rules were developed based on the training and development data. As the system converts trees to trees and does not add reentrancies, it does not produce remote edges. While the results are not among the highest-ranking in the task, the primary labeled F1 score of 71.1% in the English open track shows that a rule-based system on top of a leading dependency parser (the Stanford parser) can obtain reasonable results for this task.

9 Conclusion

The task has yielded substantial improvements to UCCA parsing in all settings. Given that the best reported results were achieved with different parsing and learning approaches than the baseline model TUPA (which has been the only available parser for UCCA), the task opens a variety of paths for future improvement. Cross-lingual transfer, which capitalizes on UCCA's tendency to be preserved in translation, was employed by a number of systems and has proven remarkably effective. Indeed, the high scores obtained for French parsing in a low-resource setting suggest that high quality UCCA parsing can be straightforwardly extended to additional languages, with only a minimal amount of manual labor.

Moreover, given the conceptual similarity between the different semantic representations (Abend and Rappoport, 2017), it is likely the parsers developed for the shared task will directly contribute to the development of other semantic parsing technology. Such a contribution is facilitated by the available conversion scripts available between UCCA and other formats.

Acknowledgments

We are deeply grateful to Dotan Dvir and the UCCA annotation team for their diligent work on the corpora used in this shared task.

This work was supported by the Israel Science Foundation (grant No. 929/17), and by the HUJI Cyber Security Research Center in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office.

References

Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *Proc. of ACL*, pages 228–238.

Omri Abend and Ari Rappoport. 2017. The state of the art in semantic representation. In *Proc. of ACL*, pages 77–89.

Omri Abend, Shai Yerushalmi, and Ari Rappoport. 2017. UCCAApp: Web-application for syntactic and semantic phrase-based annotation. *Proc. of ACL System Demonstrations*, pages 109–114.

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The parallel meaning bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. *CoRR*, abs/1702.03964.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proc. of the Linguistic Annotation Workshop*.

Alexandra Birch, Omri Abend, Ondřej Bojar, and Barry Haddow. 2016. HUME: Human UCCA-based evaluation of machine translation. In *Proc. of EMNLP*, pages 1264–1274.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

Leshem Choshen and Omri Abend. 2018. Reference-less measure of faithfulness for grammatical error correction. In *Proc. of NAACL (Short papers)*, pages 124–129.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL (to appear)*.

Robert M. W. Dixon. 2010a. *Basic Linguistic Theory: Grammatical Topics*, volume 2. Oxford University Press.

Robert M. W. Dixon. 2010b. *Basic Linguistic Theory: Methodology*, volume 1. Oxford University Press.

Robert M. W. Dixon. 2012. *Basic Linguistic Theory: Further Grammatical Topics*, volume 3. Oxford University Press.

Shibhansh Dohare and Harish Karnick. 2017. Text summarization using abstract meaning representation. *CoRR*, abs/1706.01678.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proc. of ACL*, pages 1127–1138.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018. Multitask parsing across semantic representations. In *Proc. of ACL*, pages 373–385.

Fuad Issa, Marco Damonte, Shay B. Cohen, Xiaohui Yan, and Yi Chang. 2018. Abstract meaning representation for paraphrase detection. In *Proc. of NAACL*, pages 442–452.

Wei Jiang, Yu Zhang, Zhenghua Li, and Min Zhang. 2019. HLT@SUDA at semEval-2019 Task 1: UCCA graph parsing as constituent tree parsing. In *Proc. of SemEval-2019*.

Nikita Kitaev and Dan Klein. 2018. Consituency parser with self-attentive decoder. In *ACL*, pages 2676–2686.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *Proc. of NAACL-HLT*, pages 1077–1086, Denver, Colorado.

Weimin Lyu, Sheng Huang, Abdul Rafae Khan, Shengqiang Zhang, Weiwei Sun, and Jia Xu. 2019. CUNY-PKU parser at SemEval-2019 Task 1: Cross-lingual semantic parsing with UCCA. In *Proc. of SemEval-2019*.

Gabriel Marzinotto, Johannes Heinecke, and Géraldine Damnati. 2019. MaskParse@Deskiñ at semEval-2019 Task 1: Cross-lingual UCCA semantic parsing with recursive masked sequence tagging. In *Proc. of SemEval-2019*.

Dang Tuan Nguyen and Trung Tran. 2019. DANGNT@UIT.VNU-HCM at SemEval-2019 Task 1: Graph transformation system from Stanford basic dependencies to Universal Conceptual Cognitive Annotation (UCCA). In *Proc. of SemEval-2019*.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proc. of LREC*.

Joakin Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proc. of ACL*, pages 99–106.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Zdenka Uresova. 2016. Towards comparability of linguistic graph banks for semantic parsing. In *Proc. of LREC*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*, pages 915–926.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Tobias Pütz and Kevin Glocker. 2019. Tüpa at SemEval-2019 Task 1: (Almost) feature-free semantic parsing. In *Proc. of SemEval-2019*.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *ACL*, pages 818–827.

Elior Sulem, Omri Abend, and Ari Rappoport. 2015. Conceptual annotations preserve structure across translations: A French-English case study. In *Proc. of S2MT*, pages 11–22.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018a. Semantic structural annotation for text simplification. In *NAACL 2018*, pages 685–696.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018b. Simple and effective text simplification using semantic and neural methods. In *Proc. of ACL*, pages 162–173.

Shiva Taslimipoor, Omid Rohanian, and Sara Može. 2019. GCN-Sem at SemEval-2019 Task 1: Semantic parsing using graph convolutional and recurrent neural networks. In *Proc. of SemEval-2019*.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on universal dependencies. In *Proc. of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1713–1723.

Dian Yu and Kenju Sagae. 2019. UC Davis at SemEval-2019 Task 1: DAG semantic parsing with attention-based decoder. In *Proc. of SemEval-2019*.

Sheng Zhang, Kevin Duh, and Benjamin Van Durme. 2017. Selective decoding for cross-lingual open information extraction. In *Proc. of IJCNLP*, pages 832–842.

Sheng Zhang, Xutai Ma, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2018. Cross-lingual decompositional semantic parsing. In *Proc. of EMNLP*, pages 1664–1675.

HLT@SUDA at SemEval-2019 Task 1: UCCA Graph Parsing as Constituent Tree Parsing

Wei Jiang, Zhenghua Li[*], Yu Zhang, Min Zhang
School of Computer Science and Technology, Soochow University, China
{wjiang0501, yzhang25}@stu.suda.edu.cn, {zhli13,minzhang}@suda.edu.cn

Abstract

This paper describes a simple UCCA semantic graph parsing approach. The key idea is to convert a UCCA semantic graph into a constituent tree, in which extra labels are deliberately designed to mark remote edges and discontinuous nodes for future recovery. In this way, we can make use of existing syntactic parsing techniques. Based on the data statistics, we recover discontinuous nodes directly according to the output labels of the constituent parser and use a biaffine classification model to recover the more complex remote edges. The classification model and the constituent parser are simultaneously trained under the multi-task learning framework. We use the multilingual BERT as extra features in the open tracks. Our system ranks the first place in the six English/German closed/open tracks among seven participating systems. For the seventh cross-lingual track, where there is little training data for French, we propose a language embedding approach to utilize English and German training data, and our result ranks the second place.

1 Introduction

Universal Conceptual Cognitive Annotation (UCCA) is a multi-layer linguistic framework for semantic annotation proposed by Abend and Rappoport (2013). Figure 1 shows an example sentence and its UCCA graph. Words are represented as terminal nodes. Circles denote non-terminal nodes, and the semantic relation

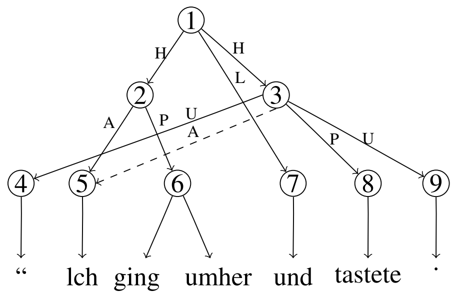


Figure 1: A UCCA graph example from the German data. The English translation is “ I went around and groped . We assign a number to each non-terminal node to facilitate illustration.

between two non-terminal nodes is represented by the label on the edge. One node may have multiple parents, among which one is annotated as the primary parent, marked by solid line edges, and others as remote parents, marked by dashed line edges. The primary edges form a tree structure, whereas the remote edges enable reentrancy, forming directed acyclic graphs (DAGs).[1] The second feature of UCCA is the existence of nodes with discontinuous leaves, known as discontinuity. For example, node 3 in Figure 1 is discontinuous because some terminal nodes it spans are not its descendants.

Hershcovich et al. (2017) first propose a transition-based UCCA Parser, which is used as the baseline in the closed tracks of this shared task. Based on the recent progress on transition-based parsing techniques, they propose a novel set of transition actions to handle both discontinuous and remote nodes and design useful features based on bidirectional LSTMs. Hershcovich et al. (2018) then extend their previous approach and propose to utilize the annotated data with other

[*]Corresponding author, hlt.suda.edu.cn/zhenghua

[1]The full UCCA scheme also has implicit and linkage relations, which are overlooked in the community so far.

semantic formalisms such as abstract meaning representation (AMR), universal dependencies (UD), and bilexical Semantic Dependencies (SDP), via multi-task learning, which is used as the baseline in the open tracks.

In this paper, we present a simple UCCA semantic graph parsing approach by treating UCCA semantic graph parsing as constituent parsing. We first convert a UCCA semantic graph into a constituent tree by removing discontinuous and remote phenomena. Extra labels encodings are deliberately designed to annotate the conversion process and to recover discontinuous and remote structures. We heuristically recover discontinuous nodes according to the output labels of the constituent parser, since most discontinuous nodes share the same pattern according to the data statistics. As for the more complex remote edges, we use a biaffine classification model for their recovery. We directly employ the graph-based constituent parser of Stern et al. (2017) and jointly train the parser and the biaffine classification model via multi-task learning (MTL). For the open tracks, we use the publicly available multilingual BERT as extra features. Our system ranks the first place in the six English/German closed/open tracks among seven participating systems. For the seventh cross-lingual track, where there is little training data for French, we propose a language embedding approach to utilize English and German training data, and our result ranks the second place.

2 The Main Approach

Our key idea is to convert UCCA graphs into constituent trees by removing discontinuous and remote edges and using extra labels for their future recovery. Our idea is inspired by the pseudo non-projective dependency parsing approach propose by Nivre and Nilsson (2005).

2.1 Graph-to-Tree Conversion

Given a UCCA graph as depicted in Figure 1, we produce a constituent tree shown in Figure 2 based on our algorithm described as follows.

1) Removal of remote edges. For nodes that have multiple parent nodes, we remove all remote edges and only keep the primary edge. To facilitate future recovery, we concatenate an extra "remote" to the label of the primary edge, indicating that the corresponding node has other remote relations. We can see that the label of the child node 5 becomes "A-remote" after conversion in Figure 1 and 2.

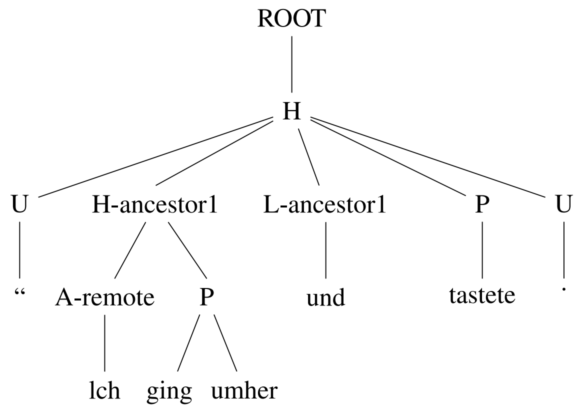


Figure 2: Constituent tree converted from UCCA gragh.

	train	dev	total	percent(%)
ancestor 1	1460	149	1609	91.3
ancestor 2	96	19	115	6.5
ancestor 3	21	0	21	1.2
discontinuous	16	2	18	1.0

Table 1: Distribution of discontinuous structures in the English-Wiki data, which is similar in the German data.

2) Handling discontinuous nodes. We call node 3 in Figure 1 a discontinuous node because the terminal nodes (also words or leaves) it spans are not continuous ("lch ging umher und" are not its descendants). Since mainstream constituent parsers cannot handle discontinuity, we try to remove discontinuous structures by moving specific edges in the following procedure.

Given a discontinuous node $A = 3$, we first process the leftmost non-descendant node $B = “lch''$. We go upwards along the edges until we find a node $C = 2$, whose father is either the lowest common ancestor (LCA) of $A = 3$ and $B = “lch''$ or another discontinuous node. We denote the father of $C = 2$ as $D = 1$.

Then we move $C = 2$ to be the child of $A = 3$, and concatenate the original edge label with an extra string (among "ancestor 1/2/3/..." and "discontinuous") for future recovery, where the number represents the number of edges between

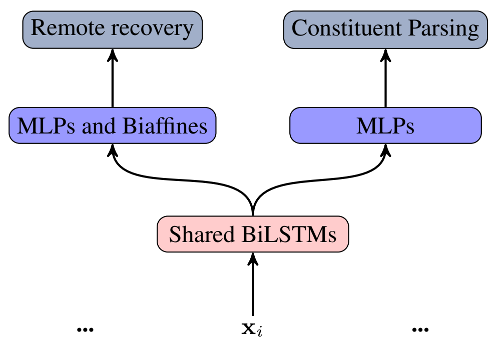


Figure 3: The framework of MTL.

the ancestor $D = 1$ and $A = 3$.

After reorganizing the graph, we then restart and perform the same operations again until there is no discontinuity.

Table 1 shows the statistics of the discontinuous structures in the English-Wiki data. We can see that D is mostly likely the LCA of A and B, and there is only one edge between D and A in more than 90% cases.

Considering the skewed distribution, we only keep "ancestor 1" after graph-to-tree conversion, and treat others as continuous structures for simplicity.

3) Pushing labels from edges into nodes. Since the labels are usually annotated in the nodes instead of edges in constituent trees, we push all labels from edges to the child nodes. We label the top node as "ROOT".

2.2 Constituent Parsing

We directly adopt the minimal span-based parser of Stern et al. (2017). Given an input sentence $\mathbf{s} = w_1...w_n$, each word w_i is mapped into a dense vector $\mathbf{x}_i$ via lookup operations.

$$\mathbf{x}_i = \mathbf{e}_{w_i} \oplus \mathbf{e}_{t_i} \oplus ...$$

where $\mathbf{e}_{w_i}$ is the word embedding and $\mathbf{e}_{t_i}$ is the part-of-speech tag embedding. To make use of other auto-generated linguistic features, provided with the datasets, we also include the embeddings of the named entity tags and the dependency labels, but find limited performance gains.

Then, the parser employs two cascaded bidirectional LSTM layers as the encoder, and use the top-layer outputs as the word representations.

Afterwards, the parser represents each span $w_i...w_j$ as

$$\mathbf{r}_{i,j} = (\mathbf{f}_j - \mathbf{f}_i) \oplus (\mathbf{b}_i - \mathbf{b}_j)$$

where $\mathbf{f}_i$ and $\mathbf{b}_i$ are the output vectors of the top-layer forward and backward LSTMs.

The span representations are then fed into MLPs to compute the scores of span splitting and labeling. For inference, the parser performs greedy top-down searching to build a parse tree.

2.3 Remote Edge Recovery

We borrow the idea of the state-of-the-art biaffine dependency parsing (Dozat and Manning, 2017) and build our remote edge recovery model. The model shares the same inputs and LSTM encoder as the constituent parser under the MTL framework (Collobert and Weston, 2008). For each remote node, marked by "-remote" in the constituent tree, we consider all other non-terminal nodes as its candidate remote parents. Given a remote node A and another non-terminal node B, we first represent them as the span representations. $\mathbf{r}_{i,j}$ and $\mathbf{r}_{i',j'}$, where i, i', j, j' are the start and end word indices governed by the two nodes. Please kindly note that B may be a discontinuous node.

Following Dozat and Manning (2017), we apply two separate MLPs to the remote and candidate parent nodes respectively, producing $\mathbf{r}^{\texttt{child}}_{i,j}$ and $\mathbf{r}^{\texttt{parent}}_{i',j'}$.

Finally, we compute a labeling score vector via a biaffine operation.

$$\mathbf{s}(A \leftarrow B) = \begin{bmatrix} \mathbf{r}^{\texttt{child}}_{i,j} \\ 1 \end{bmatrix}^{\mathrm{T}} \mathbf{W}\mathbf{r}^{\texttt{parent}}_{i',j'} \quad (1)$$

where the dimension of the labeling score vector is the number of the label set, including a "NOT-PARENT" label.

Training loss. We accumulate the standard cross-entropy losses of all remote and non-terminal node pairs. The parsing loss and the remote edge classification loss are added in the MTL framework.

2.4 Use of BERT

For the open tracks, we use the contextualized word representations produced by BERT (Devlin et al., 2018) as extra input features.[2] Following previous works, we use the weighted summation of the last four transformer layers and then multiply a task-specific weight parameter following (Peters et al., 2018).

[2]We use the multilingual cased BERT from https://github.com/google-research/bert.

3 Cross-lingual Parsing

Because of little training data for French, we borrow the treebank embedding approach of Stymne et al. (2018) for exploiting multiple heterogeneous treebanks for the same language, and propose a language embedding approach to utilize English and German training data. The training datasets of the three languages are merged to train a single UCCA parsing model. The only modification is to concatenate each word position with an extra language embedding (of dimension 50), i.e. $\mathbf{x}_i \oplus \mathbf{e}_{lang=en/de/fr}$ to indicate which language this training sentence comes from. In this way, we expect the model can fully utilize all training data since most parameters are shared except the three language embedding vectors, and learn the language differences as well.

4 Experiments

Except BERT, all the data we use, including the linguistic features and word embeddings, are provided by the shared task organizer (Hershcovich et al., 2019). We also adopt the averaged F1 score as the main evaluation metrics returned by the official evaluation scripts (Hershcovich et al., 2019).

We train each model for at most 100 iterations, and early stop training if the peak performance does not increase in 10 consecutive iterations.

Table 2 shows the results on the dev data. We have experimented with different settings to gain insights on the contributions of different components. For the single-language models, it is clear that using pre-trained word embeddings outperforms using randomly initialized word embeddings by more than 1% F1 score on both English and German. Finetuning the pre-trained word embeddings leads to consistent yet slight performance improvement. In the open tracks, replacing word embedding with the BERT representation is also useful on English (2.8% increase) and German (1.2% increase). Concatenating pre-trained word embeddings with BERT outputs leads is also beneficial.

For the multilingual models, using randomly initialized word embeddings is better than pre-trained word embeddings, which is contradictory to the single-language results. We suspect this is due to that the pre-trained word embeddings are independently trained for different languages and would lie in different semantic spaces without proper aligning. Using the BERT outputs is tremendously helpful, boosting the F1 score by more than 10%. We do not report the results on English and German for brevity since little improvement is observed for them.

Methods	F1 score		
	Primary	Remote	Avg
Single-language models on English			
random emb	0.778	0.542	0.774
pretrained emb (no finetune)	0.790	0.494	0.785
pretrained emb	0.794	0.535	**0.789**
bert	0.821	0.593	0.817
pretrained emb ⊕ bert	0.825	0.603	**<u>0.821</u>**
official baseline (closed)	0.745	0.534	0.741
official baseline (open)	0.753	0.514	0.748
Single-language models on German			
random emb	0.817	0.549	0.811
pretrained emb (no finetune)	0.829	0.544	0.823
pretrained emb	0.831	0.536	**0.825**
bert	0.842	0.610	0.837
pretrained emb ⊕ bert	0.849	0.628	**<u>0.844</u>**
official baseline (closed)	0.737	0.46	0.731
official baseline (open)	0.797	0.587	0.792
Multilingual models on French			
random emb	0.688	0.343	0.681
pretrained emb	0.673	0.174	0.665
bert	0.796	0.524	**<u>0.789</u>**
official baseline (open)	0.523	0.016	0.514

Table 2: Results on the dev data.

5 Final Results

Table 3 lists our final results on the test data. Our system ranks the first place in six tracks (English/German closed/open) and the second place in the French open track. Note that we submitted a wrong result for the French open track during the evaluation phase by setting the wrong index of language, which leads to about 2% drop of averaged F1 score (0.752). Please refer to (Hershcovich et al., 2019) for the complete results and comparisons.

6 Conclusions

In this paper, we describe our system submitted to SemEval 2019 Task 1. We design a simple UCCA semantic graph parsing approach by making full use of the recent advance in syntactic parsing community. The key idea is to convert UCCA graphs into constituent trees. The graph recovery

Tracks	F1 score		
	Primary	Remote	Avg
English-Wiki_closed	0.779	0.522	0.774
English-Wiki_open	0.810	0.588	0.805
English-20K_closed	0.736	0.312	0.727
English-20K_open	0.777	0.392	0.767
German-20K_closed	0.838	0.592	0.832
German-20K_open	0.854	0.641	0.849
French-20K_open	0.779	0.438	0.771

Table 3: Final results on the test data in each track. Please refer to the official webpage for more detailed results due to the limited space

problem is modeled as another classification task under the MTL framework. For the cross-lingual parsing track, we design a language embedding approach to utilize the training data of resource-rich languages.

Acknowledgements

The authors would like to thank the anonymous reviewers for the helpful comments. We also thank Chen Gong for her help on speeding up the minimal span parser. This work was supported by National Natural Science Foundation of China (Grant No. 61525205, 61876116).

References

Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *Proc. of ACL*, pages 228–238.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. of ICML*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of ICLR*.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for ucca. In *Proc. of ACL*, pages 1127–1138.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018. Multitask parsing across semantic representations. In *Proc. of ACL*, pages 373–385.

Daniel Hershcovich, Zohar Aizenbud, Leshem Choshen, Elior Sulem, Ari Rappoport, and Omri Abend. 2019. Semeval 2019 task 1: Cross-lingual semantic parsing with ucca. *arXiv:1903.02953*.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proc. of ACL*, pages 99–106.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proc. of ACL*, pages 818–827.

Sara Stymne, Miryam de Lhoneux, Aaron Smith, and Joakim Nivre. 2018. Parser training with heterogeneous treebanks. In *Proc. of ACL*, pages 619–625.

SemEval-2019 Task 2: Unsupervised Lexical Frame Induction

Behrang QasemiZadeh
SFB991, Germany
zadeh@phil.hhu.de

Miriam R. L. Petruck
ICSI, US
miriamp@icsi.berkeley.edu

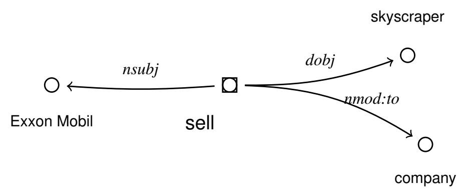
Regina Stodden
HHUD, Germany
stodden@phil.hhu.de

Laura Kallmeyer
HHUD, SFB991, Germany
kallmeyer@phil.hhu.de

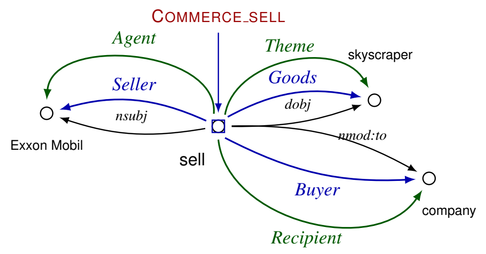
Marie Candito
Paris Diderot University - CNRS, France
marie.candito@linguist.univ-paris-diderot.fr

Abstract

This paper presents Unsupervised Lexical Frame Induction, Task 2 of the International Workshop on Semantic Evaluation in 2019. Given a set of prespecified syntactic forms in context, the task requires that verbs and their arguments be clustered to resemble semantic frame structures. Results are useful in identifying polysemous words, i.e., those whose frame structures are not easily distinguished, as well as discerning semantic relations of the arguments. Evaluation of unsupervised frame induction methods fell into two tracks: Task A) *Verb Clustering* based on FrameNet 1.7; and B) *Argument Clustering*, with B.1) based on FrameNet's core frame elements, and B.2) on VerbNet 3.2 semantic roles. The shared task attracted nine teams, of whom three reported promising results. This paper describes the task and its data, reports on methods and resources that these systems used, and offers a comparison to human annotation.



(a) Input: subcategorization frames.



(b) Output: Semantic Frame Tagging using labels learned by Unsupervised methods.

Figure 1: Given semantically unlabeled structures (1a), annotate the input with semantic information learned via unsupervised methods (1b).

1 Introduction

SemEval 2019 Task 2 focused on the unsupervised semantic labeling of a set of prespecified (semantically) unlabeled structures (Figure 1). Unsupervised learning methods analyze these structures (Figure 1a) to augment them with semantic labels (Figure 1b). The shape of the manually labeled input frames is constrained to an *acyclic* connected tree of lexical items (words and multi-word units) of maximum depth 1, where just one root governs several arguments. The task used Berkeley FrameNet (FN) (Ruppenhofer et al., 2016) and Q. Zadeh and Petruck (2019), guidelines for this task, to determine the arguments and label them with semantic information.

We compared the proposed system results for unsupervised semantic tagging with that of human annotated (or, gold-standard) data in three different subtasks (Figure 2). To evaluate the systems, we computed distributional similarities between their generated unsupervised labeled data and human annotated reference data. For computing similarities we used general purpose numeral methods of text clustering, in particular BCUBED F-SCORE (Bagga and Baldwin, 1998) as the single figure of merit to rank the systems.

The most important result of the shared task is the creation of a benchmark for a future complex task. This benchmark includes a moderately sized, manually annotated set of frames, where only the verbs of each were included, along with their *core frame elements* (which uniquely define a frame as Ruppenhofer et al. describe). To complement FN's core frame elements that have highly specific meanings, the benchmark also includes the annotated argument structures of the verbs based on the generic semantic roles proposed for verb classes in VerbNet 3.2 (Kipper et al., 2000; Palmer et al., 2017). The benchmark comes with simplified annotation guidelines and a modular annotation system with browsing and editing capabilities.[1] Complementing the benchmarking are several state-of-the-art competing baselines, from the participants, that serve as a point of departure for improvements in the future.[2]

The rest of this paper is organized as follows: Section 2 contextualizes this task; Section 3 offers a detailed task-description; Section 4 describes the data; Section 5 introduces the evaluation metrics and baselines; Section 6 characterizes the participating systems and unsupervised methods that participants used; Section 7 provides evaluation scores and additional insight about the data; and Section 8 presents concluding remarks.

2 Background

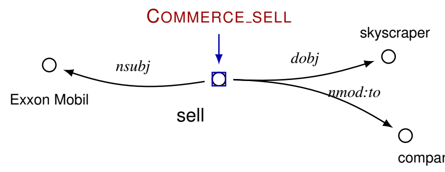
Frame Semantics (Fillmore, 1976) and other theories (Gamerschlag et al., 2014) that adopt typed feature structures for representing knowledge and linguistic structures have developed in parallel over several decades in theoretical linguistic studies about the syntax–semantics interface, as well as in empirical corpus-driven applications in natural language processing. Building repositories of (lexical) semantic frames is a core component in all of these efforts. In formal studies, lexical semantic frame knowledge bases instantiate foundational theories with tangible examples, e.g., to provide supporting evidence for the theory. Practically, frame semantic repositories play a pivotal role in natural language understanding and semantic parsing, both as inspiration for a representation format and for training data-driven machine learning systems, which is required for tasks such as information extraction, question-answering, text summarization, among others.

However, manually developing frame semantic databases and annotating corpus-derived illustrative examples to support analyses of frames are resource-intensive tasks. The most well-known frame semantic (lexical) resource is FrameNet (Ruppenhofer et al., 2016), which only covers a (relatively) small set of the vocabulary of contemporary English. While NLP research has integrated FrameNet data into semantic parsing, e.g., Swayamdipta et al. (2018), these methods cannot extend beyond previously seen training labels, tagging out-of-domain semantics as *unknown* at best. This limitation does not hinder unsupervised methods, which will port and extend the coverage of semantic parsers, a common challenge in semantic parsing (Hartmann et al., 2017).

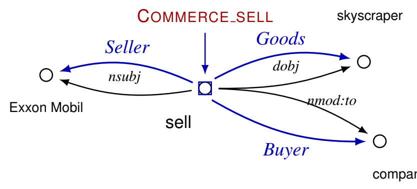
Unsupervised frame induction methods can serve as an assistive semantic analytic tool, to build language resources and facilitate linguistic studies. Since the focus is usually to build language resources, most systems (Pennacchiotti et al. (2008); Green et al. (2004)) have used a lexical semantic resource like WordNet (Miller, 1995) to extend coverage of a resource like FrameNet. Some methods, e.g., Modi et al. (2012) and Kallmeyer et al. (2018), tried to extract FrameNet-like resources automatically without additional semantic information. Others (Ustalov et al. (2018); Materna (2012)) addressed frame induction only for verbs with two arguments.

Lastly, unsupervised frame induction methods can also facilitate linguistic investigations by capturing information about the reciprocal relationships between statistical features and linguistic or extra-linguistic observations (e.g., Reisinger et al. (2015)). This task aimed to benchmark a class of such unsupervised frame induction methods.

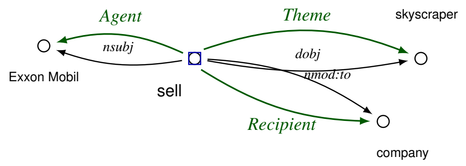
3 Task Description



(a) **Task A** - Identifying Semantic Frames: Unsupervised learned labels evaluated against FN's lexical units



(b) **Task B.1** - Full Frame Semantic Tagging: Unsupervised labels evaluated against FN's frames



(c) **Task B.2** – Case Role Labeling: Unsupervised labels evaluated against generic semantic roles (VerbNet)

Figure 2: Subtasks of SemEval 2019 Task 2.

[1] http://sfa.phil.hhu.de/.

[2] See https://competitions.codalab.org/competitions/19159 for accessing the task's language resources, tools, and further technical details.

The ambitious goal of this task was the unsupervised induction of frame semantic structures from tokenized and morphosyntacally labeled text corpora. We sought to achieve this goal by building an evaluation benchmark for three tasks. **Task A** dealt with unsupervised labeling of verb lemmas with their frame meaning. **Task B** involved unsupervised argument role labeling, where **B.1** benchmarked unsupervised labeling of frame-specific frame elements (FEs) based on FN, and **B.2** benchmarked unsupervised role labeling of arguments in Case Grammar terms (Fillmore, 1968) and against a set of generic semantic roles, taken primarily from VerbNet.

The task was unsupervised in that it forbade the use of any explicit semantic annotation (only permitting morphosyntactic annotation). Instead, we encouraged the use of unsupervised representation learning methods (e.g., word embeddings, brown clusters) to obtain semantic information. Hence, systems learn and assign semantic labels to test records without appealing to any explicit training labels. For development purposes, developers received a small labeled development set.

3.1 Task A: Clustering Verbs

The goal of this task was to identify verbs that evoke the same frame. The task involved labeling verb uses in context to resemble their categorization based on Frame Semantics (Figure 2a). Here, we used FN 1.7 as the reference for frame definitions. Hence, the task constituted the unsupervised induction of FN's lexical units, where a lexical unit (LU) is a pairing of a lemma and a frame. For example, we expected that the LUs *auction*.v, *retail*.v, *sell*.v, etc., which evoke the typed situation of COMMERCE_SELL, be labeled with the same unsupervised tag.[3]

The task resembles word sense induction in that it assigns a class (or sense) label to a verb. In word sense induction (WSI), labels are determined and evaluated on word forms (lemma + part-of-speech e.g., *sell.v* or *auction.n*). WSI evaluations assume that the inventory of senses (set S_is) for different word forms f is devised independently. For instance, assuming f_1 is labeled with the set of senses S_1 and f_2 with S_2, then $S_1 \cap S_2 \neq \phi$ only if $f_1 = f_2$; and, if $f_1 \neq f_2$ then $S_1 \cap S_2 = \phi$ (as in other SemEval benchmarks, including Agirre and Soroa (2007); Manandhar et al. (2010); Jurgens and Klapaftis (2013); Navigli and Vannella (2013)). For instance, in WSI evaluations based on OntoNotes (Hovy et al., 2006), six different labels from S_{sell} are assigned to the lemma $sell$.v, and one label s' is assigned to $auction$.v, knowing that $s' \notin S_{sell}$. Typically, lexical semantic relationships among members of S_is (e.g., synonymy, antonymy) are then analyzed independently of WSI (e.g., Lenci and Benotto (2012); Girju et al. (2007); McCarthy and Navigli (2007)). In contrast, this task assumes that the sense inventory is defined independent of word forms.

This task involves uncovering mapping between word forms f and members of S such that different word forms (i.e., $f_i \neq f_j$) can be mapped to the same meaning (label), and the same meaning (label) can be mapped to several word forms. We defined S with respect to FrameNet and assumed that its typed-situation frames are units of meaning. So, COMMERCE_SELL captures the meaning associated with both $sell$.v and $auction$.v., as well as other selling-related words. Hence, in some sense, Task A goes beyond the ordinary WSI task as it also demands identifying (unspecified) lexical semantic relationships between verbs.

3.2 Task B.1: Unsupervised Frame Semantic Argument Labeling

Taking the frames as primary and defining roles relative to each frame, the aim of Task B.1 was to cluster prespecified verb-headed argument structures according to the principles of Frame Semantics, where FrameNet served as the reference for evaluation. This task amounted to unsupervised labeling of frames and core FEs (Figure 2b). Because FrameNet defines FEs frame-specifically, Task B.1 entails Task A.

Given a set of semantically-unlabelled arguments as input (e.g., Figure 1a), the root nodes (i.e., verbs) are clustered and assigned to a set of unsupervised frame labels π_i ($1 \leq i \leq n$, where n is the number of latent frames). Then, the arguments are labeled with semantic role labels (FEs) interpreted locally given the frame. That is, for any pair of π_x and π_y, the set of assigned roles R_x to arguments under π_x are assumed to be independent from R_y labels for π_y ($R_x \cap R_y = \phi$).

3.3 Task B.2: Unsupervised Case Role Labeling

We defined Subtask B.2 in parallel to Subtask B.1 and involved an idea from Case Grammar. The ar-

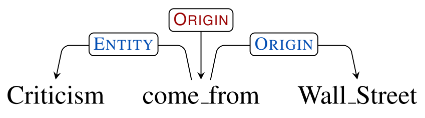
[3]Dark red small caps indicate FN frames.

guments of a verb in a set of prespecified subcategorization frames were clustered according to a common set of generic semantic roles (Figure 2c). Here, the task assumed that semantic roles are universal and generic (e.g., Agent, Patient). Their configuration determines the argument structure of verb-headed phrases. We evaluated this unsupervised labeling of arguments with semantic roles independently of the class, sense, and word form of a verb. We compared the role labels against a set of semantic roles from VerbNet 3.2 (Kipper et al., 2000). Given a verb instance, no guarantee exists that input argument structures for B.2 and B.1 would be the same.

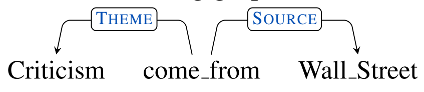
4 Evaluation Dataset

The dataset consists of manual annotations for verb-headed frame structures anchored in tokenized sentences. These frame structures were manually annotated using the guidelines for this task (Q. Zadeh and Petruck, 2019). For example, as already illustrated, the verb *come_from*.v is annotated in terms of FN's ORIGIN frame and its *core* FEs, as Example 1 shows.

(1) Criticism of futures COMES FROM Wall Street.



Also, using the set of 32 generic semantic role labels in VerbNet 3.2 and two additional roles, COGNIZER and CONTENT, we annotated arguments of the verb as the following graphic shows.



We assumed unique identifiers for sentences, e.g., #s1 for Example 1. The evaluation record for *come_from*.v (Task A) appears below, where #s1 4_5 specifies the position of the verb in the sentence (Example 1).

A [#s1 4_5 come_from.ORIGIN]

Similarly, for Task B.1 and Task B.2, respectively, the evaluation records are as follows here.

B.1 [#s1 4_5 come_from.ORIGIN Criticism-:-1-:-ENTITY Wall_Street-:-6_7-:-ORIGIN]

B.2 [#s1 4_5 come_from.NA Criticism-:-1-:-THEME Wall_Street-:-6_7-:-SOURCE]

We stripped off the manually asserted labels from the records and passed them to systems for assigning unsupervised labels. Evidently, later a scorer program (Section 5) compared system-generated labels with the manually assigned labels.

4.1 Data Sampling

We sampled data from the *Wall Street Journal* (WSJ) corpus of the Penn Treebank. Kallmeyer et al. (2018) provided frame annotations similar to those in this task for a portion of WSJ sentences, using SemLink (Bonial et al., 2013) and EngVallex (Cinková et al., 2014) to generate frame semantic annotations semi-automatically. That work was based on FrameNet and the Prague Dependency Treebank (PSD) (Hajič et al., 2012) from the Broad-coverage Semantic Dependency resource (Oepen et al., 2016). We started by annotating a portion of the records in Kallmeyer et al. (2018), and later deviated from this subset to create a more representative sample of the overall diversity and distribution of verbs in the WSJ corpus using a stratified random sampling method.

4.2 Guidelines

The annotation guidelines for this task were slightly different from those of FrameNet and various semantic dependency treebanks. In contrast to FN, which annotates a full span of text as an argument filler, or PropBank, which annotates syntactic constituents of arguments of verbs (Palmer et al., 2005), we identified the text spans and only annotated a single word or a multi-word unit (MWU), i.e., the semantic head of the span, like annotations in Oepen et al. (2016) and Abstract Meaning Representation (Banarescu et al., 2013). To illustrate, in Example 1, FN would annotate *Criticism of futures* as filling the FE ENTITY. We only annotated *Criticism*, understanding it as the LU that evokes JUDGMENT_COMMUNICATION, which in turn represents the meaning of the whole text span. Thus, we assumed that another frame f_a fills an argument of a frame. We annotated only the main content word(s) that evoke(s) f_a; these main words are the *semantic heads*.[4]

Multi-word unit semantic heads (e.g., named entities, word form combinations) are annotated as if a single word form, such as *Wall Street* (# 1), excluding modifiers. In contrast to semantic depen-

[4]The annotation guidelines (Q. Zadeh and Petruck, 2019) discuss decisions about marking semantic heads and the complex situations resulting from it for argument annotation.

dency structures (e.g., DELPH-IN MRS-Derived Semantic Dependencies, Enju PredicateArgument Structures, and Tectogramatical Representation in PSD (Oepen et al., 2016)), we did not commit to the underlying syntactic structure of the sentence since we were not obliged to relabel only syntactic structures. Rather, we annotated words and MWUs if the frame analysis permitted doing so.[5]

4.3 Annotation Procedure

We annotated the data in a modular manner and in a semi-controlled environment using an annotation system developed for this purpose. The procedure consisted of four steps: 1) Reading and Comprehension; 2) Choosing a Frame; 3) Annotating Arguments; and 4) Rating, Commenting, or Revising. We tracked and logged all changes in the data as well as annotator interaction with the annotation system upon starting to annotate. The tool measured the time that annotators spent on each record and each annotation step, as well as how annotators moved between steps.

In Step 1, annotators viewed a sentence with one highlighted verb, as in Example 2.

(2) Criticism of futures **COMES** from Wall Street.

The goal of this step was understanding the meaning of the verb and its semantic function, and identifying semantic heads of arguments and their associated words or MWUs. To continue, an annotator must confirm the understanding of the verb's meaning of the verb, and can identify its semantic arguments. Without confirmation, an annotator would terminate the annotation process for that input sentence and go to the next one.

If confirmed, Step 2 required the annotator to choose the frame that the verb evoked. This step may have included annotating multi-word phrasal verbs, e.g., COMES+FROM (Example 2). The annotation system assisted by providing a list of likely frames for the verb, including a LU lookup function (as in FN), an extended set of LUs derived via statistical methods, and previously logged annotations. After reviewing the definitions of the proposed frames, annotators chose one, or annotated the verb form with a different existing FN frame. Otherwise, the annotator terminated the process and the record moved to the list of "skipped items".

The annotation of arguments, Step 3, required that annotators label the core FEs of the chosen frame by first identifying their semantic head, which first may have required marking MWUs, e.g., Wall+Street in Example 3, below.

(3) Criticism of futures **comes from** Wall Street.

The tool lists the core FEs and their definitions, and checks the integrity of record annotations to ensure that each core FE is annotated only once. In parallel, annotators add the verb's subcategorization frame and its semantic role. We did not annotate null instantiated FEs (but FN does). During step 3, annotators could go back to the previous step and change their choice of frame type.

For Step 4, annotators rated their annotation, stating their opinion on how well the annotated instance fit FrameNet's definition and how it compared to other annotated instances. In a sense, annotators measured their confidence in the assigned labels. They did so by selecting a number on a scale from 1 to 5, with 1 not confident at all and 5 the most confident, i.e., the annotation fit perfectly to the chosen FrameNet frame, its definition, and examples. Annotators had the option to add free text comments on each record.

The annotation procedure was rarely straightforward. Given the interdependence of Steps 2 and 3, annotators usually moved back and forth between them. In Step 2 an annotator might believe that a target verb did not belong in any existing FN frame. Likewise, annotators could terminate the annotation process even upon reaching the last step.

4.3.1 Quality Control

At least two annotators verified all annotation used in the evaluation. A main annotator annotated all records in the dataset; two other annotators verified or disputed those annotations. If annotators could not reach an agreement, we removed the record from the SemEval dataset.

A full analysis of annotator disagreement goes beyond the scope of this work. While the source of annotator disagreement may seem trivial and simple (e.g., only one annotator understood the sentence correctly), we believe that some sentences may have more than one interpretation, all of which are plausible. Like the disagreement resulting from incorrect frame assignment, deciding what frame a verb evokes may be challenging; and resolving the dilemma is not always simple. Choosing between two related frames (e.g.,

[5]Q. Zadeh and Petruck describe the issues in detail.

BUILDING vs. INTENTIONALLY_CREATE, related via Inheritance in FN), or identifying metaphorical and non-metaphorical uses of a verb requires subtle and sophisticated understanding of the semantics of the language, and of Frame Semantics. At times, disagreements pointed to more complex linguistic issues that remain in debate, e.g., choosing the semantic head of a syntactically complex argument, treating quantifiers, conjunctions, etc.

4.4 Summary statistics

Table 1 shows a statistical summary of the annotation task. The **SemEval** column reports the statistics for the final set of records, i.e., gold records with double-agreement between annotators, and which we used to evaluate the systems. **Total** reports the statistics of all analyzed records, from which we chose our **SemEval** data. **Skipped** and **InProg** show the statistics for discarded records and records without a final decision, respectively. **Dev** shows the statistics for the development set.

Each of the rows reports a value of a component of the data or annotator interaction with the data. **Records** indicates the number of annotated verbs and their arguments. **Sentences** and **Tokens** indicate the size of the sub-corpus of the annotated records. **VF** is the number of distinct verb lemmas (273), mapped to the number of distinct frames that the **Frames-Type** row shows (149) (Figure 3 in Appendix A.1 plots their frequency distribution.) **FElements** reports the number of annotated FEs categorized under the number of FE types shown in the **FE-Type** row. **Sem-Arg** shows the number of annotated verb arguments with VerbNet-like semantic roles, classified into 32 of 41 possible semantic role categories. **Multi-word** lists the number of annotated MWUs

	SemEval	Total	Skipped	InProg	Dev
Records	4,620	5,637	301	716	594
Sentences	3,346	3,803	294	675	582
Tokens	90,460	102,067	8,329	19,151	15198
Verb-Forms	273	373	93	210	35
Frame-Type	149	234	75	185	37
#FEs	9,510	11,269	373	1,386	1,128
FE-Type	198	270	64	197	62
Sem-Arg	9,466	11,215	370	1,379	1,079
Multi-word	2,366	2,773	61	346	368
Confidence	3.30	3.2	2.41	2.5	3.34
Time	539h	742h	25h	177h	19h
Total-Move	68,784	83,753	1,903	13,066	4,406

Table 1: Annotation and Data Statistical Summary

Confidence reports the average of annotator-assigned confidence scores for annotations per record. Although interpreting this measure demands more work, the averages appear to be as expected. Specifically, **SemEval** is higher in value than both **InProg** and **Skipped**, facts that we associate with double agreement and the choice reviewing process. Still, many records with high confidence scores remained as **InProg** given the lack of double agreement. Table 5 (Appendix A.1) lists the top 10 frames annotated with their respective highest and lowest confidence ratings averaged by their frequency in **SemEval**.

The last two rows of Table 1 are meta-data on the annotation process. **Time** reports the total time annotators spent in active annotation, engaged in the steps described above (742 hours), excluding the reviewing process (Section 4.3.1) and including the time to annotate MWUs. **Total-Move** is the total number of logical moves for frame annotation between annotators and the annotation system, i.e., logged changes in the process of frame and core FE annotation. This number excludes annotation of verb subcategorization with generic semantic roles.[6]

In **SemEval**, annotated frames had an average of 2.15 arguments, requiring a minimum of five logical moves to annotate (MWU-less sentences). However, on average, each **SemEval** record required 14.8 moves. This number is even higher for **InProg** (18.2); we believe that it indicates the complexity of the annotation task. Table 4 (Appendix A.1) further details annotator activity, with time spent and moves per annotation step. As expected, frame annotation of verbs (Step 2), was the most time consuming part of the task.

4.5 Development Dataset

Shared task participants received a development set consisting of 600 records from a total of 4,620 records, where Table 4 shows the statistics. The development set contained gold annotations for all three subtasks.

5 Evaluation Metrics

For all subtasks, as figure of merit, here we report the performance of participating systems with measures for evaluating text clustering techniques, including the classic measures of Purity (PU), inverse-Purity (IPU), and their harmonic mean (PIF) (Steinbach et al., 2000), as well as the harmonic mean for BCubed precision and recall (i.e.,

[6]With the exception of a few verbs, annotators rarely changed the annotation system's rule-based suggestions of VerbNet semantic roles.

BcP, BcR, and BcF, respectively) (Bagga and Baldwin, 1998).

To compute these measures for the pairing of reference-labeled data and unsupervised-labeled data (with each having an exact set of annotated items), we built a contingency table T with rows for gold labels and columns for unsupervised system labels. We filled the table with the number of intersecting items, as done in cross-tabulation of results in classification tasks to compute precision and recall. For Task A (Section 3), T tracks the unsupervised system labels and the gold reference labels assigned to verbs. For Task B.1, we labeled the rows and columns of T with tuples (l_v, l_a), where l_v labels the frame evoking verb and l_a labels the FE filler. For Task B.2, the rows and columns in T track the unsupervised system labels and the gold reference labels (generic semantic roles) assigned to arguments.

These performance measures reflect a notion of similarity between the distribution of unsupervised labels and that of the gold reference labels, given certain criteria. Specifically, they define the notions of consistency and completeness of automatically generated clusters based on the evaluation data. Each method measures consistency and completeness in its own way, and alone may lack sufficient information for a clear understanding and analysis of system performance (Amigó et al., 2009). But, as the *single metric for system ranking*, we used the BcF measure, given its satisfactory behavior in certain situations. Note that we modeled the task and its evaluation as hard clustering, where a record receives only one label, without overlap in any generated category of items.

5.1 Baselines

Similar to other clustering tasks, we use baselines of random, all-in-one-cluster (AIn1), and one-cluster-per-instance (1CpI). Additionally, we adapted the baseline of *the most frequent sense* in WSI for these tasks by introducing the one-cluster-per-head (1CpH) baseline in Task A, and one-cluster-per-syntactic-category (1CpG) for verb argument clustering in Task B.2.[7] For Task B.1, we built a baseline, 1CpGH for labeling verbs with their lemmas (as in 1CpH) and FEs with grammatical relation to their heads (as in 1CpG). We included two more labels lcmpx and rcmpx for frame fillers with no direct syntactic relation to the head verb, if occurring left of or right of the verb, respectively.

Both 1CpH and 1CpG (and their combination for Task B.1) are hard to beat because of the long-tailed distribution of the frequency of our test data. E.g., most verbs frequently instantiate one particular frame and rarely other ones. Similarly, a particular role (FE) frequently is filled by words that have a particular grammatical relation to its governing verb; e.g., most subjects of most verb forms receive the agent label in their subcategorization frame (or, an agent-like element in their Frame Semantics representations). Evidently the chosen labels for grammatical relations influences 1CpG and 1CpHG scores. Values reported later (specifically, Tables 6 and 2) could be improved by employing heuristics, e.g., relabeling enhanced dependencies using a few rules.

We also employed one unsupervised and a second supervised system baselines. For the unsupervised one, we trained the system with data from Kallmeyer et al. (2018). For the supervised one, we used OPEN-SESAME, a state-of-the-art supervised FrameNet tagger (Swayamdipta et al., 2018). After converting its output to the format of the present task, we evaluated it similar to other systems. Both systems were trained out-of-the-box with no additional tuning.

6 System Descriptions

We received submissions from nine teams (13 participants). Only three chose to submit system description papers. Arefyev et al. (2019) provided a solution for Task A and Task B.2, using both sets of these results to address Task B.1. Task A used language models and Hearst-like patterns to tune and obtain contextualized vector representations for the verbs in the test set. A hierarchical agglomerative clustering method followed, where hyperparameters were set with labeled and unlabeled records from the development and test sets. Task B.2 employed a logistic regression trained over the development set to identify only the most frequent labels. The classifier was based on features obtained from a language model and hand-crafted rules. Using logistic regression and training this algorithm with the development set remains an issue of concern, given the intended unsupervised scenario. While we objected to using the development set to train a supervised system for this

[7]We use syntactic dependencies of the Enhanced Universal Dependencies formalism (Schuster and Manning, 2016).

subtask, we still report its scores. The differences between its results and those of the other systems may be informative. Still, we considered Arefyev et al.'s results for Task B only complementarily, not to rank the systems.

Anwar et al. (2019) proposed a method that was similar to that of Arefyev et al. (2019). Arefyev et al. used contextualized word embeddings from the BERT language modeling tool Devlin et al. (2018), whereas Anwar et al. used pre-trained embeddings. They merged the outputs of Tasks A and B.2 for Task B.1. Task A used agglomerative clustering of vectors with concatenated verb representation vectors and vectors that represent usage context. Task B.2 employed hand crafted features, a method to encode syntactic information, and again an agglomerative clustering method.

Ribeiro et al. (2019) also reported results for all subtasks using similar techniques to those reported in the other two submitted papers. Ribeiro et al. (2019) used the bidirectional neural language model BERT, which Arefyev et al. (2019) also used. Task A employed contextualized word representations proposed in (Ustalov et al., 2018), and Biemann's clustering algorithm (Biemann, 2006). Compared to the two other systems, Ribeiro et al. (2019) exploited input structures, weighted them, and used them elegantly in its algorithm. With the same method but different hyper-parameters for B.2 along with combining results from Task A, Ribeiro et al. (2019) offered a solution to B.1.

7 Results and Data Analysis

Table 2 reports the BcF scores for system submissions along with a baseline for each task.[8] As the table shows, each system performs best only in one of the tasks. We report Arefyev et al.'s submission for Tasks B.1 and B.2 only to show the benefit of using a small amount of training data and a supervised method together with a clustering algorithm, provided that such training data is available. As readers know, finding the optimal (actual) number of clusters is an open research area. Participants knew the number of clusters: whereas Arefyev et al. and Anwar et al. used this information, Ribeiro et al. opted for a statistical method tuned with data that we provided.

The baseline systems, the unsupervised method of Kallmeyer et al. (2018) performed the worst

System	BcF	BcF	BcF
Arefyev et al.	**70.70**	~~63.12~~	~~64.09~~
Anwar et al.	68.10	**49.49**	42.1
Ribeiro et al.	65.32	42.75	**45.65**
BASELINE	65.35	45.79	39.03
	Task A	B.1	B.2

Table 2: Summary of Results. The BASELINE for Task A is 1CPH, and for B.1 and B.2 is 1CPHG. Best results appear in bold face; discarded results are crossed out. Table 6 lists all other baselines.

of all systems regarding BcF. This result is not surprising since that work did not effectively handle MWUs in the test, where only the head of the MWU was kept. However, the output of Open-SESAME, and its low BcF was indeed surprising.

We fed Open-SESAME the sentences from the test set; it identified approximately 5k frames. However, the overlap with the test set was only 1,216 records (identification problem in Open-SESAME). These 1,216 records exhibit a mismatch between 536 of the arguments and their respective target verbs. We ignored the system's extra or incorrectly generated arguments, and replaced the missing items with those of the 1CPHG baseline records. We then used the resulting records for evaluation against the task's gold data as did the task's participants. As Table 3 shows, the unsupervised method outperforms the supervised system for all tasks by a wide margin, i.e.,the unsupervised label set can carry more information than does the supervised label set.

	BcP	BcR	BcF
Task A	84.52	44.67	58.45
Task B.1	81.04	31.6	45.47
Task B.2	34.26	36.56	35.37

Table 3: Open-SESAME Performance

We compared results for confidence measure that annotators assigned to records. First, we split the evaluation records according to their assigned confidence value into five subsets E_i, $1 \leq i \leq 5$, such that subset E_1 contained only records with confidence value 1, E_2 contained only record with confidence value 2, etc.. Then we evaluated system outputs on each subset E_i and logged that BcF. Later, we performed this evaluation cumulatively using subsets E_i's by adding records from all E_js to E_i where $i < j$. Interpreting the obtained values requires careful attention (e.g., changes in the prior probabilities of gold clusters and their

[8]The full list of baselines and performance measures appear in Table 6 of the Appendix.

cardinality must be taken into account), overall, we observed a similar trend for all systems: as expected, namely a positive correlation between the confidence value and BCF. Thus, what human annotators usually found hard to annotate, automatic systems also found hard to cluster. (The reverse relation does not hold). Or, pessimistically, the level of noise in annotation increases as their associated confidence decreases. (Table 7 in Appendix A.2 details the results.)

Finally, we wanted to identify the frames that machines found difficult to cluster. To estimate difficulty we used the differences in BCF under the following conditions. We repeated the evaluation process $1 \leq i \leq n$ times (where n is the number of gold labels for a task) for each system. In each iteration i, we removed all data items of a gold category i. We measured and noted the resulting BCF in the given iteration; we deduced the score from the system performance over the entire gold set. To cancel frequency effects, we normalized the differences by the number of gold data instances. We removed all records annotated as COMMERCE_SELL from the evaluation set E to form E'. We computed the BCF of the systems over E' ($E' \subset E$), and measured $d = E_{\mathrm{BCF}} - E'_{\mathrm{BCF}}$. We interpreted a positive difference as an easy to cluster gold category i, and a negative difference as a hard to cluster gold category i.

The heat maps in Table 8 and Table 9 show a summary of the results for Task A and Task B.2, respectively. All systems performed similarly for approximately 30% of the gold classes. Comparing differences across systems and the baselines of 1CPH and 1CPG reveals (possibly) interesting information. Thus, for example, in Task A, most systems found COMMERCE_SELL hard and COMMERCE_BUY easy to cluster. Interestingly, a set of six verbs evokes each frame: *buy, purchase, buy_back, buy_up, buy_out, buy_into* for COMMERCE_BUY; and *sell, retail, auction, place, deal, resell* for COMMERCE_SELL. From these two sets of verbs, three are polysemous: *buy* in the former, and *place* and *deal* in the latter. Does the morphology of the verbs (e.g., *buy-back*, *resell*) make one easy to cluster? Alternatively, are other factors at play, such as the number of verb instances? How these factors might influence the proposed naive BCF-difference model is an open question.

8 Concluding Remarks

We have presented the SemEval 2019 task on unsupervised lexical frame induction. We described the task in detail, provided a summary of methods that participants developed, and compared the results. Although much room for improvement of the task remains, we consider it a step forward. It employed a well-motivated typology of lexical frames to distinguish lexical frame induction tasks. The evaluation data derived from annotations of a well-known resource, namely a portion of WSJ sentences, perhaps the most annotated corpus of English. These features provide opportunities for future investigation, in particular in studies related to reciprocal relations between syntactic and lexical semantic frame structures.

One reason to promote using unsupervised methods is their inherent flexibility to embrace unknown data. These methods have a high margin of tolerance for *noise*, and perform better than supervised method with insufficient training data. For unsupervised data, obtaining or generating training data is easier than doing so with supervised methods because they simply do not require annotation. For example, all participant systems could collect similar unlabeled training data from only syntactically annotated corpora to generate more unlabeled records. Ultimately, such methods can achieve respectable performance, and produce clusters which are both more informative than the unlabeled input and supervised categories (under certain situations). As shown, unsupervised methods can even outperform a state-of-the-art Frame Semantics parser by a wide margin (Section 7), while a very large gap remains for improvements in future work.

Acknowledgements

This research was funded by DFG - SFB991. We thank Timm Lichte, Rainer Oswald, Curt Anderson, and Kurt Erbach. We also thank the LDC for its generous support, and the NVIDIA Corporation for the Titan Xp GPU used in this work.

References

Eneko Agirre and Aitor Soroa. 2007. Semeval-2007 task 02: Evaluating word sense induction and discrimination systems. In *Proceedings of the 4th International Workshop on Semantic Evaluation*, SemEval '07, pages 7–12, Stroudsburg, PA, USA. Association for Computational Linguistics.

Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. 2009. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Inf. Retr.*, 12(4):461–486.

Saba Anwar, Dmitry Ustalov, Nikolay Arefyev, Simone Paolo Ponzetto, Chris Biemann, and Alexander Panchenko. 2019. Hm^2 at semeval 2019 task 2: Unsupervised frame induction using contextualized and uncontextualized word embeddings. In *Proceedings of The 13th International Workshop on Semantic Evaluation*.

Nikolay Arefyev, Boris Sheludko, Adis Davletov, Dmitry Kharchev, Alex Nevidomsky, , and Alexander Panchenko. 2019. Neural granny at semeval 2019 task 2: A combined approach for better modeling of semantic relationships in semantic frame induction. In *Proceedings of The 13th International Workshop on Semantic Evaluation*.

Amit Bagga and Breck Baldwin. 1998. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 1*, COLING '98, pages 79–85, Stroudsburg, PA, USA. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186. Association for Computational Linguistics.

Chris Biemann. 2006. Chinese whispers - an efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of TextGraphs: the First Workshop on Graph Based Methods for Natural Language Processing*, pages 73–80. Association for Computational Linguistics.

Claire Bonial, Kevin Stowe, and Martha Palmer. 2013. Renewing and revising semlink. In *Proceedings of the 2nd Workshop on Linked Data in Linguistics (LDL-2013): Representing and linking lexicons, terminologies and other language data*, pages 9 – 17, Pisa, Italy. Association for Computational Linguistics.

Silvie Cinková, Eva Fučíková, Jana Šindlerová, and Jan Hajič. 2014. EngVallex - English valency lexicon. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

C. J. Fillmore. 1976. Frame Semantics and the Nature of Language. *Annals of the New York Academy of Sciences*, 280(Origins and Evolution of Language and Speech):20–32.

Charles J. Fillmore. 1968. The case for case. In *Universals in Linguistic Theory*, pages 1–88. Holt Rinehart and Winston, New York.

Thomas Gamerschlag, Doris Gerland, Rainer Osswald, and Wiebke Petersen, editors. 2014. *General Introduction*. Springer International Publishing, Cham.

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2007. Semeval-2007 task 04: Classification of semantic relations between nominals. In *Proceedings of the Fourth International Workshop on Semantic Evaluation (SemEval-2007)*, pages 13–18. Association for Computational Linguistics.

Rebecca Green, Bonnie J. Dorr, and Philip Resnik. 2004. Inducing frame semantic verb classes from WordNet and LDOCE. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing prague czech-english dependency treebank 2.0. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*. European Language Resources Association (ELRA).

Silvana Hartmann, Ilia Kuznetsov, Teresa Martin, and Iryna Gurevych. 2017. Out-of-domain framenet semantic role labeling. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 471–482.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, NAACL-Short '06, pages 57–60, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Jurgens and Ioannis Klapaftis. 2013. Semeval-2013 task 13: Word sense induction for graded and non-graded senses. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 290–299.

Laura Kallmeyer, Behrang QasemiZadeh, and Jackie Chi Kit Cheung. 2018. Coarse lexical frame acquisition at the syntax–semantics interface using a latent-variable pcfg model. In *Proceedings of the Seventh*

Joint Conference on Lexical and Computational Semantics, pages 130–141, New Orleans, Louisiana. Association for Computational Linguistics.

Karin Kipper, Hoa Trang Dang, and Martha Palmer. 2000. Class-based construction of a verb lexicon. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 691–696. AAAI Press.

Alessandro Lenci and Giulia Benotto. 2012. Identifying hypernyms in distributional semantic spaces. In *SemEval 2012*, pages 75–79. Association for Computational Linguistics.

Suresh Manandhar, Ioannis Klapaftis, Dmitriy Dligach, and Sameer Pradhan. 2010. Semeval-2010 task 14: Word sense induction & disambiguation. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 63–68, Uppsala, Sweden. Association for Computational Linguistics.

Jiří Materna. 2012. Lda-frames: An unsupervised approach togenerating semantic frames. In *Computational Linguistics and Intelligent Text Processing*, pages 376–387, Berlin, Heidelberg. Springer Berlin Heidelberg.

Diana McCarthy and Roberto Navigli. 2007. Semeval-2007 task 10: English lexical substitution task. In *Proceedings of the Fourth International Workshop on Semantic Evaluation (SemEval-2007)*, pages 48–53. Association for Computational Linguistics.

George A. Miller. 1995. WordNet: A lexical database for English. *Commun. ACM*, 38(11):39–41.

Ashutosh Modi, Ivan Titov, and Alexandre Klementiev. 2012. Unsupervised induction of frame-semantic representations. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 1–7, Montréal, Canada. Association for Computational Linguistics.

Roberto Navigli and Daniele Vannella. 2013. Semeval-2013 task 11: Word sense induction and disambiguation within an end-user application. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 193–201, Atlanta, Georgia, USA. Association for Computational Linguistics.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Zdenka Uresova. 2016. Towards comparability of linguistic graph banks for semantic parsing. In *LREC 2016*, Paris, France. ELRA.

Martha Palmer, Claire Bonial, and Jena Hwang. 2017. Verbnet: Verbnet: Capturing english verb behavior, meaning, and usage. In *The Oxford Handbook of Cognitive Science*. Oxford Press.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Comput. Linguist.*, 31(1):71–106.

Marco Pennacchiotti, Diego De Cao, Roberto Basili, Danilo Croce, and Michael Roth. 2008. Automatic induction of framenet lexical units. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 457–465, Stroudsburg, PA, USA. Association for Computational Linguistics.

Behrang Q. Zadeh and Miriam R. L. Petruck. 2019. Guidelines for the semantic frame annotation system. corpus annotation guidelines TR.9.2018, SFB991 - ICSI.

Drew Reisinger, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van Durme. 2015. Semantic proto-roles. *Transactions of the Association for Computational Linguistics*, 3:475–488.

Eugénio Ribeiro, Vânia Mendonça, Ricardo Ribeiro, David Martins de Matos, Alberto Sardinha, Ana Lúcia Santos, and Luísa Coheur. 2019. L2F/INESC-ID at SemEval-2019 Task 2: Unsupervised Lexical Semantic Frame Induction using Contextualized Word Representations. In *Proceedings of The 13th International Workshop on Semantic Evaluation*.

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck, Christopher R. Johnson, Collin F. Baker, and Jan Scheffczyk. 2016. *FrameNet II: Extended Theory and Practice*. ICSI, Berkeley.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English Universal Dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

M. Steinbach, G. Karypis, and V. Kumar. 2000. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*.

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A. Smith. 2018. Syntactic scaffolds for semantic structures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3772–3782, Brussels, Belgium. Association for Computational Linguistics.

Dmitry Ustalov, Alexander Panchenko, Andrey Kutuzov, Chris Biemann, and Simone Paolo Ponzetto. 2018. Unsupervised semantic frame induction using triclustering. In *ACL*, pages 55–62, Melbourne, Australia. ACL.

A Appendices

A.1 Appendix I: Annotation Process

A.1.1 Time and Moves per Annotation Step

Table 4 shows the amount of effort to develop the SemEval dataset in terms of time and moves that the annotation system recorded. (See Sections 4.3, 4.4).

Annotator Activity	**Time**	**Moves**
Reading and Comprehension	78	4,795
Choosing a Frame	177	9,737
Annotating Arguments	81	19,510
Rating, Revising, Commenting	115	25,793
Multi-word Unit Annotation	89	8,949
Total	539	68,784

Table 4: Total hours and number of moves for each annotation step for the 4,620 record dataset.

A.1.2 Plot of frequency of annotated frames

Figure 3 plots the frequency distribution of the annotated frames in the gold data (**SemEval**).

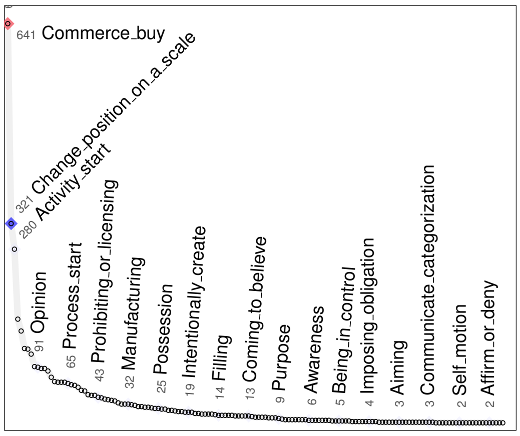


Figure 3: Frequency Distribution of Annotated Frames

A.1.3 Some Frames and their Averaged Confidence

Table 5 lists FN frames annotated with the highest and lowest confidence. Table 4 details hours spent to derive the evaluation data set. Section 4.3 discusses both tables. The full list of annotations in human readable form is available to browse and comment on at http://corpora.phil.hhu.de/fi/frames.html.

A.2 Appendix II: Statistical Summary of Evaluation and System Submissions

A.2.1 Unabridged Results Table

Table 6 extends Table 2. Section 5 defines the abbreviations. A horizontal line separates participating systems and the baselines.

A.2.2 Confidence Measures and BCF Performance

Table 7 shows system BCF scores for confidence. The table shows changes in the BCF of systems when altering the evaluation set based on the assigned confidence for an annotated record. (See Section 7 for an explanation).

Frame Type	**#VF**	**#Rec**	**Conf**
DECIDING	1	13	4.31
AGREE_OR_REFUSE_TO_ACT	1	15	4.13
TAKE_PLACE_OF	1	11	4
BEING_EMPLOYED	1	6	4
STATEMENT	8	149	3.97
TAKING_SIDES	3	16	3.88
ACTIVITY_STOP	4	16	3.88
COMMERCE_SELL	6	168	3.82
BRINGING	1	5	3.8
GIVE_IMPRESSION	4	39	3.79

(a) Frames with Highest Average Confidence

Frame Type	**#VF**	**#Rec**	**Conf**
BEING_IN_CONTROL	2	5	1.6
COMING_TO_BE	2	5	1.8
OPERATING_A_SYSTEM	2	10	1.8
AWARENESS	1	6	1.83
REMOVING	3	8	1.88
INTENTIONALLY_CREATE	6	19	1.95
CERTAINTY	1	68	2.03
OPINION	2	91	2.1
THWARTING	2	22	2.32
FIRST_RANK	1	21	2.38

(b) Frames with Lowest Average Confidence

Table 5: Frame types with the highest (5a) and the lowest (5b) confidence (**Conf**) by number of records (**#Rec**) with double annotator agreement. **#VF** reports the number of distinct verb forms that evoke a frame.

System	#C	PU	IPU	PIF	BCP	BCR	BCF
Arefyev et al.	272	**78.68**	77.62	**78.15**	**70.86**	70.54	**70.7**
Anwar et al.	150	72.4	**81.49**	76.68	62.17	**75.27**	68.1
Ribeiro et al.	222	72.84	77.84	75.25	61.25	69.96	65.32
Kallmeyer et al.	218	73.77	72.86	73.31	64.62	65.48	65.05
1CPI	4620	100	3.23	6.25	100	3.23	6.25
AIN1	1	13.87	100	24.37	3.78	100	7.28
1CPH	273	82.16	66.95	73.78	75.98	57.33	65.35
RANDOM	149	15.11	5.78	8.36	6.76	3.85	4.9

Task A

System	#C	PU	IPU	PIF	BCP	BCR	BCF
Arefyev et al.	~~776~~	~~72.47~~	~~72.16~~	~~72.31~~	~~62.73~~	~~63.51~~	~~63.12~~
Anwar et al.	338	55.74	**67.79**	**61.18**	43.22	**57.9**	**49.49**
Ribeiro et al.	518	52.29	57.56	54.8	39.43	46.69	42.75
Kallmeyer et al.	1023	**72.24**	49.12	58.48	**62.71**	37.51	46.94
1CPI	9510	100	4.58	8.77	100	4.58	8.77
AIN1	1	6.55	100	12.3	1.56	100	3.08
1CPHG	1203	78.46	45.99	57.99	71.11	33.77	45.79
RANDOM	436	11.34	6.04	7.88	6.03	4.81	5.35

Task B.1

System	#C	PU	IPU	PIF	BCP	BCR	BCF
Arefyev et al.	~~14~~	~~73.94~~	~~81.4~~	~~77.49~~	~~56.25~~	~~74.46~~	~~64.09~~
Anwar et al.	2	50.43	**80.47**	**62.00**	29.58	**73.00**	42.1
Ribeiro et al.	7	58.25	71.4	64.16	36.88	59.91	**45.65**
Kallmeyer et al.	37	**61.44**	51.53	56.05	**40.89**	37.33	39.03
1CPG	37	61.44	51.53	56.05	40.89	37.33	39.03
1CPI	9466	100	0.34	0.67	100	0.34	0.67
AIN1	1	34.34	100	51.13	21.66	100	35.6
RANDOM	32	34.65	4.75	8.36	21.89	3.45	5.96

Task B.2

Table 6: Complete System Results and Baselines

Cnf	#I	Arefyev	Anwar	Ribeiro
1	4620	70.7	68.10	65.32
2	4334	71.87	69.28	66.57
3	3657	74.64	72.22	70.17
4	2542	76.46	73.82	73.43
5	84	86.14	84.65	85.13

Task A

Cnf	#I	Arefyev	Anwar	Ribeiro
1	9,510	63.12	49.52	42.75
2	9017	64.20	50.44	43.61
3	7,606	67.18	53.40	46.42
4	5,356	68.70	55.99	49.20
5	169	85.16	81.85	65.60

Task B.1

Cnf	#I	Arefyev	Anwar	Ribeiro
1	9,466	64.09	42.12	45.65
2	8,911	64.98	42.32	46.27
3	7,528	66.47	42.67	47.52
4	5,292	65.71	40.67	46.95
5	167	77.19	55.18	56.58

Task B.2

Cumulative

Cnf	#I	Arefyev	Anwar	Ribeiro
1	286	73.79	70.57	67.70
2	677	66.45	63.80	60.46
3	1,115	76.71	75.98	70.01
4	2,458	76.65	74.05	73.45
5	84	86.14	84.65	85.13

Task A

Cnf	#I	Arefyev	Anwar	Ribeiro
1	493	68.57	55.37	51.84
2	1,411	59.86	49.08	42.16
3	2,250	70.67	57.97	47.60
4	5,187	68.70	56.01	49.24
5	169	85.16	81.85	65.60

Task B.1

Cnf	#I	Arefyev	Anwar	Ribeiro
1	553	52.69	39.82	38.21
2	1,385	58.36	40.99	41.55
3	2,236	69.01	48.07	49.4
4	5,125	65.44	40.37	46.72
5	167	77.19	55.18	56.58

Task B.2

Stratified

Table 7: Changes in BCF score of systems relative to changes in evaluation records based on assigned confidence measure.

A.3 Examining Clusters by Removing One Gold Cluster at a Time

#Rmvd	Frame	Arefyev	Anwar	Ribeiro	Kallmeyer	1CPH
168	Commerce_sell	🟦	🟦	⬜	🟦	🟦
6	Awareness	🟦	🟦	🟥	🟦	🟦
89	Assessing	🟦	🟦	🟦	🟦	🟦
54	Seeking_to_achieve	🟥	🟥	🟥	🟦	🟦
30	Activity_ongoing	🟦	🟦	⬜	🟦	🟦
19	Intention._creat.	🟥	🟥	🟦	🟦	🟦
2	Storing	🟥	🟦	🟥	🟦	🟦
3	Cause_..._progress	⬜	⬜	⬜	🟦	⬜
27	Process_continue	🟦	🟦	🟥	🟦	🟦
14	Filling	⬜	⬜	⬜	🟦	🟦
121	Causation	🟥	🟥	🟥	🟦	🟦
59	Hiring	🟦	⬜	⬜	🟦	🟦
89	Choosing	🟦	⬜	⬜	⬜	🟦
6	Being_employed	🟥	🟥	🟥	⬜	⬜
7	Criminal_investig.	🟥	🟥	🟥	🟥	🟥
65	Process_start	🟥	🟥	🟥	🟥	🟥
4	Notifi..._charges	⬜	⬜	🟦	🟥	🟥
28	Assistance	🟥	🟥	🟥	🟥	🟥
21	First_rank	🟥	🟥	🟥	🟥	🟥
20	Giving	🟥	🟥	🟥	🟥	🟥
2	Sentencing	🟥	🟥	🟥	🟥	🟥
16	Activity_stop	🟥	🟥	🟥	🟥	🟥
641	Commerce_buy	🟥	🟥	🟥	🟥	⬜
42	Have_associated	🟥	🟥	🟥	🟥	🟥
13	Request	🟥	🟥	🟥	🟥	🟥
149	Statement	🟥	🟥	⬜	🟥	🟥
67	Avoiding	🟥	🟥	🟥	🟥	🟥
2	Performer..._roles	🟥	🟥	🟥	🟥	🟥
2	Attack	🟥	🟥	🟥	🟥	🟥
92	Cause_change...	🟥	🟥	🟥	🟥	🟥
9	Purpose	🟥	🟥	🟥	🟥	🟥
14	Collaboration	🟥	🟥	🟥	🟥	🟥
13	Coming_to_believe	🟥	🟥	🟥	🟥	🟥

Table 8: Task A – Part of a heat map from results (Section 7), with cases that exhibit a range of difference values. Red denotes a positive and blue a negative difference; white means no change (zero difference). Differences (normalized by cluster size) are in domain 0.01 to −0.01.

#Rmvd	Frame	Arefyev	Anwar	Ribeiro	Kallmeyer	1CPH
2	Attending	🟦	🟥	⬜	🟦	🟦
38	Getting	🟦	🟦	🟦	🟦	🟦
16	Activity_stop	🟦	🟦	🟦	🟦	🟦
63	Becoming_a_membe	🟦	🟦	⬜	🟦	🟦
641	Commerce_buy	🟦	🟦	🟦	🟦	🟦
6	Retaining	⬜	🟦	🟦	🟦	🟦
7	Criminal_investigati	⬜	🟥	🟦	🟦	🟦
67	Avoiding	🟦	🟦	🟦	🟦	🟦
46	Scrutiny	🟦	🟦	⬜	🟦	🟦
280	Activity_start	🟦	🟦	🟦	🟦	🟦
5	Bringing	⬜	🟥	🟥	🟦	🟦
2	Change_event_time	🟦	🟦	🟦	🟦	🟦
89	Choosing	⬜	⬜	⬜	⬜	⬜
8	Removing	🟥	🟥	🟥	⬜	⬜
13	Coming_to_believe	⬜	🟦	⬜	⬜	⬜
14	Inspecting	⬜	⬜	⬜	⬜	⬜
3	Cause_to_end	🟥	⬜	🟦	⬜	⬜
3	Communicate_catego	🟥	🟥	🟥	⬜	⬜
13	Deciding	🟦	🟦	⬜	⬜	⬜
2	Attack	🟥	🟥	🟥	🟥	🟥
2	Creating	🟥	🟥	🟥	🟥	🟥
19	Intentionally_act	🟥	🟥	🟥	🟥	🟥
34	Cause_to_amalgamat	🟥	🟥	🟥	🟥	🟥
19	Intentionally_create	🟥	🟥	🟥	🟥	🟥
5	Usefulness	🟥	🟥	🟥	🟥	🟥
16	Taking_sides	🟥	🟥	🟥	🟥	🟥
4	Notification_of_charg	🟥	🟥	🟥	🟥	🟥
3	Aiming	🟥	🟥	🟥	🟥	🟥
121	Causation	🟥	🟥	🟥	🟥	🟥
3	Process_end	🟥	🟥	🟦	🟥	🟥
2	Transfer	⬜	⬜	🟥	🟥	🟥
5	Coming_to_be	🟥	🟥	🟥	🟥	🟥
10	Cotheme	🟥	🟥	🟥	🟥	🟥
7	Hearsay	🟥	🟥	🟥	🟥	🟥
16	Transition_to_state	🟥	🟥	🟥	🟥	🟥
92	Cause_change_of_pos	🟥	🟥	🟥	🟥	🟥
7	Inclusion	🟥	🟥	🟥	🟥	🟥
7	Simultaneity	🟥	🟥	🟥	🟥	🟥
4	Imposing_obligation	🟥	🟥	🟥	🟥	🟥
3	Cause_change	🟥	🟥	🟥	🟥	🟥
3	Distributed_position	🟥	🟥	🟥	🟥	🟥
39	Give_impression	🟥	🟥	🟥	🟥	🟥
4	Supporting	🟥	🟥	🟥	🟥	🟥

Table 9: Heat map that visualizes Task B.2 data

Neural GRANNy at SemEval-2019 Task 2: A combined approach for better modeling of semantic relationships in semantic frame induction

Nikolay Arefyev[1,2], **Boris Sheludko**[1,2], **Adis Davletov**[1,2], **Dmitry Kharchev**[1,2], **Alex Nevidomsky**[1], and **Alexander Panchenko**[3,4]

[1]Samsung R&D Institute Russia, Moscow, Russia
[2]Lomonosov Moscow State University, Moscow, Russia
[3]Skolkovo Institute of Science and Technology, Moscow, Russia
[4]Language Technology Group, University of Hamburg, Hamburg, Germany

Abstract

We describe our solutions for semantic frame and role induction subtasks of SemEval 2019 Task 2. Our approaches got the highest scores, and the solution for the frame induction problem officially took the first place. The main contributions of this paper are related to the semantic frame induction problem. We propose a combined approach that employs two different types of vector representations: dense representations from hidden layers of a masked language model, and sparse representations based on substitutes for the target word in the context. The first one better groups synonyms, the second one is better at disambiguating homonyms. Extending the context to include nearby sentences improves the results in both cases. New Hearst-like patterns for verbs are introduced that prove to be effective for frame induction. Finally, we propose an approach to selecting the number of clusters in agglomerative clustering.

1 Introduction

Semeval-2019 Task 2 consisted of three subtasks, this paper presents solutions to all three which were all performing better than other submitted approaches. The first solution officially took the first place in the competition, the other two used tuning on the development set provided by the organizers, which was then interpreted as using additional corpora.

Semantic Frame Induction (Subtask A) is the task of grouping target word occurrences in a text corpus according to their frame (meaning and semantic arguments structure). Target words are usually verbs, nouns, and adjectives (these have argument structure; however in the shared task dataset only verbs were present). For instance, the verbs *rise, fall* and *climb* in the sentences *The dollar is rising, which makes Russian economy unstable* and *The dollar fell 1% in September after climbing 2% in August* should be clustered together, while the verb *climb* in sentences like *People climb mountains* should be clustered separately. For the sake of brevity, occurrences of different words sharing the same frame will be called synonyms, and occurrences of the same word belonging to different frames will be called homonyms. This may violate the traditional meaning of these terms. For instance, *fall* and *rise* are not considered synonyms in the classical sense. Semantic Role Induction refers to finding realizations of semantic arguments in text and relating them to corresponding semantic frame slots. Generic role induction (subtask B.2) requires a small number of frame-independent roles like *Agent, Patient, Theme*, etc. Frame-specific role induction (subtask B.1) allows labeling arguments of each frame independently from other frames. For instance, *Microsoft* in *Microsoft bought Github* and *Google* in *Google opened new offices* should be labeled as the same role in B.2 but may be labeled differently in B.1. For further details please refer to QasemiZadeh et al. (2019).

In this paper, we focused mainly on the Frame Induction subtask. The main contributions for this subtask are the following. A combined approach to semantic frame induction is introduced, which clusters dense representations obtained from hidden layers of a masked LM first and sparse bag-of-words representations of possible substitutes for a word in context afterward. This approach resulted in better clustering of both synonyms and homonyms[1]. New Hearst-like patterns designed specifically for verbs were used and they proved to be beneficial for Semantic Frame Induction. Also, a simple but effective semi-supervised approach to selecting the number of clusters for agglomerative clustering was proposed. Finally, we proposed ex-

[1]GRANNy in the team name stands for General Relation Acquisition with Neural Networks

tending context with neighboring sentences which have shown consistent improvements for both of our representations. For solving subtask B.2 we used a semi-supervised approach of training logistic regression over features that were partly designed and partly learned in an unsupervised fashion. To ensure the best performance on verbs that were not present in training data (the majority of examples in the test) we used cross-validation with a lexical split, to select optimal features and hyperparameters. For solving subtask B.1 we trivially reused labels from B.2

2 Related Work

This section describes previous work which our approach is based on. Word Sense Induction (WSI) is the task of clustering occurrences of an ambiguous word according to their meaning which is similar to Frame Induction. One of the major differences from Frame Induction is that WSI doesn't require grouping together different words with similar meanings, however, we adopt some ideas from WSI in this work. Instead of graph or vector representation of word co-occurrence information traditionally used to solve WSI task, Baskaya et al. (2013) proposed exploiting n-gram language model (LM) to generate possible substitutes for an ambiguous word in a particular context. Their approach was one of the best in SemEval-2013 WSI shared task (Jurgens and Klapaftis, 2013). Struyanskiy and Arefyev (2018) proposed pretraining SOTA neural machine translation model built from Transformer blocks (Vaswani et al., 2017) to restore target words hidden from its input (replaced with a special token CENTERWORD). After pretraining, they exploited both predicted output embeddings to represent ambiguous words and attention weights to better weigh relevant context words in word2vec weighted average representation. A combination of these representations achieved SOTA results on one of the datasets from RUSSE'2018 Word Sense Induction for the Russian language shared task (Panchenko et al., 2018). Amrami and Goldberg (2018) develop ideas from Baskaya et al. (2013) exploiting neural bidirectional LM ELMO (Peters et al., 2018) instead of n-gram LM for generating substitutes. To improve results further they propose using dynamic symmetric patterns "T and _", "_ and T" (here "T" stands for the target word and "_" for the position at which we collect LM predictions). For instance, to represent the word *orange* in *He wears an orange shirt* instead of predicting what comes after *wears* in *He wears* they predict what comes after *and* in *He wears orange and* (similarly, for backward LM they predict what comes before *and* in *and orange shirt*). This provides more information to the LM because we don't hide the ambiguous word and forces it to produce its co-hyponyms instead of all possible continuations given a one-sided context. Other important contributions include lemmatizing substitutes to remove grammatical bias from representations (which was especially important for verbs) and using IDF weights to penalize frequent substitutes, which are probably worse for discriminating between senses. They achieve SOTA results on the SemEval-2013 WSI dataset.

Devlin et al. (2018) proposed BERT (Bidirectional Encoder Representations from Transformers). Like the model from Struyanskiy and Arefyev (2018), BERT is a deep NN built from Transformer blocks and pretrained on the task of restoring words hidden from its input (replaced with a special token [MASK], hence they named it masked LM). However they used much deeper models, pretrained them on much more data and predicted hidden words at each timestep rather than generating them as an output sequence. Also additional next sentence prediction task was used to pretrain the model for sentence pairs classification (like paraphrase detection and NLI). BERT has shown better results than previous SOTA models on a wide spectrum of natural language processing tasks.

3 Semantic Frame Induction

In this section, we describe our approaches to building vector representations of an occurrence of the target word (which is always a verb in the SemEval-2019 Frame Induction task dataset). The first approach exploits dense vector representations of the target word in a context obtained from hidden layers of BERT model. Another approach builds sparse TF-IDF BOW vectors from substitutes generated for the target word by BERT masked LM. We found that each model has its own downsides when used with non-trainable distance functions like cosine and Euclidean, and with traditional clustering algorithms like agglomerative clustering, DBScan, and affinity propagation. The first approach didn't discriminate different senses of the same verb, the second one had problems with clustering together similar senses of different verbs. In preliminary experiments, we tried fixing the first problem by learning a distance function instead of using a fixed one, but this didn't help, presumably due to a very small amount of labeled data provided and restrictions on using additional labeled data. So our best performing algorithm is two-stage: it groups examples to a relatively small number of large clusters using the first representation (merging synonyms together while not taking into consideration homonyms) and then splits each of them into smaller clusters using the second representation (disambiguating homonyms). Finally, we describe our approach to clustering these vector representations and propose a technique for selecting the appropriate number of clusters.

3.1 BERT Hidden Representations

In the preliminary experiments, we compared dense representations from different layers of two BERT models pretrained on English texts: bert-base-uncased and bert-large-uncased with 3x more weights. While being significantly slower, the large model didn't show better clustering results for the development set, so we stuck to the base model. Presumably, fine-tuning the large model to the final task could reveal its superiority, but this would require much more labeled data that was provided. Interestingly, a weighted average of word2vec embeddings for context words proposed for WSI in Arefyev et al. (2018) showed similar results, which also supports the hypothesis that distance functions like cosine or Euclidean are not appropriate for BERT hidden representations. BERT-base consists of 12 Transformer blocks with 12 attention heads each, hidden state dimensionality is 768. It was pre-trained on lowercased texts split into subword units. Hyperparameters were selected on the development set, the best results were achieved using outputs of the layer 6 at timestep when the first subword of the verb was fed in. Also, better results were achieved when input texts were lemmatized. This can be explained by the large grammatical bias of LMs also noticed by Amrami and Goldberg (2018): it is much easier to correctly predict grammatical attributes like number, gender, tense from contexts, so it is more beneficial to assign higher probabilities to all verbs with correct tense than to all verbs with correct meaning when losses like cross-entropy are used, which results in large distance between occurrences of the same verb in the same meaning, but in different tenses.

3.2 Substitutes Representations

We adopt ideas from Amrami and Goldberg (2018) for our second approach to Frame Induction, with several important differences. First, we propose new patterns which are more suitable for verbs. Secondly, we use BERT, which proved to be better than ELMO for generating substitutes in our series of preliminary experiments. This is likely due to the fact that BERT takes into account the whole context in all of its layers, unlike bidirectional LM in ELMO, which consists of two independently trained language models, one using only right context, and another only left context. Lastly, we do hard clustering instead of soft clustering required for SemEval-2013 WSI, hence we do not sample from distributions predicted by LM, but instead, take the topmost probable substitutes. We found this approach works better than one doing soft clustering and then selecting the most probable cluster for each example.

To generate substitutes, a masked LM based on the bert-base-uncased model was utilized. It is likely that the large model could generate better substitutes, but we left it for future work. Non-lemmatized lowercased text was passed through all the layers of the model. We didn't add biases of the last linear layer to obtain less frequent but more contextually suitable subwords. We took K most probable substitutes to represent each example (K=40 was selected on the development set), lemmatized them to get rid of grammatical bias, and then built TF-IDF bag-of-words vectors. To improve results we employ symmetric patterns. Symmetric patterns were first proposed in Hearst (1992) and then used in many cases, including Widdows and Dorow (2002), Panchenko et al. (2012), Schwartz et al. (2015), to extract lexical relations like hyponymy, hypernymy, co-hyponymy, etc. from texts, and to augment lexical resources. However, we were not aware of any Hearst-like patterns designed specifically for verbs. Along with "T and _" pattern and trivial "T" and "_" patterns we proposed and experimented with "T and then _", "T and will _" and "T and then will _" patterns. We suppose that the meaning of a verb is better described not by its hypernyms or co-hyponyms (which are traditionally extracted for nouns using patterns like "_ such as T" or "T and _") but rather by preceding and following events which are better extracted by the proposed patterns. "T and then _" pattern has shown the best results both for the development and the test sets. For instance, to generate substitutes for the verb *build* in *They are building phones* we pass *They are building and then* [MASK] *phones* and collect predictions at the masked timestep. We found that among others, substitutes like *export, distribute, ship* are generated for *Manufacturing* frame and *establish, open, close* for *Building* frame of the verb *build* allowing to discriminate between them. See Appendix A for examples.

3.3 Clustering

We experimented with K-means, DBScan, Affinity Propagation and Agglomerative clustering algorithms implemented in the scikit-learn (Pedregosa et al., 2011) and found agglomerative clustering to achieve the best results. To select hyperparameters of Agglomerative clustering for dense representations (number of clusters and distance functions between points and clusters) we used a simple yet effective semi-supervised approach: merge the development and test sets (labeled and unlabeled respectively) and perform grid search for hyperparameters that provide clustering with optimal value of the target metric (BCubed-f1 in our case) on the labeled subset. Almost always optimal results were obtained using cosine distance for points and average linkage for clusters (average distance between elements).

3.4 Combined Approach

Our best performing submission was made of a combination of techniques described above. At phase 1, we clustered dense representations using proposed semi-supervised agglomerative clustering. At phase 2, we split each cluster separately using sparse representations and conventional agglomerative clustering with cosine distance and average linkage (selected on the development set). We didn't use the semi-supervised tuning again because at that stage most clusters didn't contain labeled examples. During the blind evaluation period, we simply split each cluster into two (this method is denoted as **Combined** below). In the post-evaluation period, we experimented with more sophisticated approaches. Finally, our best results (denoted as *Combined2*) were obtained when the number of clusters at phase 2 was selected using silhouette score and small clusters (with less than 20 examples) or clusters with different target verbs were left intact. Also, during the post-evaluation period, we tried extending the context with nearby sentences (sentences with adjacent IDs in the Penn Treebank corpus). This allowed us to incorporate more information about the preceding and following events, which resulted in improved performance of both representations. In **Combined2** we passed a large context of maximum 7 sentences to the left and to the right for dense, and smaller context of 2 sentences on both sides for sparse representations (selected on the development set).

3.5 Dataset and Experiments

Due to limitations imposed by the task, we restricted ourselves to only using labeled data provided by the organizers. For the majority of our experiments, we used the development set that consisted of 600 examples of 35 verbs clustered into 41 frames. There are many examples of synonymy in this dataset but not so many of homonymy. Almost all ambiguous verbs have less than 5 examples for all frames except their most frequent frame, hence we used only verbs *join* and *believe* (54/9 and 12/8 examples of their first/second most frequent frame respectively) to select hyperparameters likely resulting in a suboptimal performance on the test.

For internal evaluation of different representations and hyperparameters selection, we used the following procedure: the development set or its subset was clustered many times using agglomerative clustering with all feasible hyperparameter values, and maximum BCubed-f1 value (maxB3f1) was taken as a score for the representation. This allowed us to compare clusterability of different representations while avoiding problems of selecting the number of clusters and other hyperparameters. Of course, there is a possibility that other clustering algorithms might perform better with different representations, however, we didn't see improvements from using other clustering algorithms and stick to agglomerative clustering. Table 1 shows maxB3f1 for the whole development set and for all examples of several homonyms. Evidently, dense representations are significantly better when clustering the whole development set, while sparse representations with

	dev	join@dev	build@test	follow@test	start@test
sparse	0.91	0.98	0.83	0.96	0.75
dense	0.94	0.92	0.70	0.80	0.72

Table 1: Sparse vs. dense representations, maxB3f1

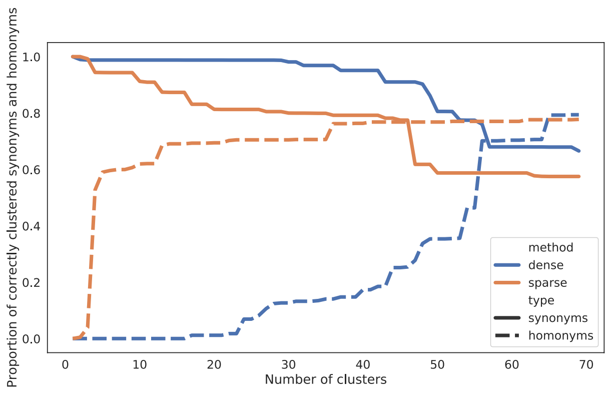


Figure 1: Recall for synonyms and homonyms w.r.t. number of clusters for dense and sparse representations

Method	#cl	PuIpuF1	B3P	B3R	B3F1
Verb baseline	227	73.94	74.61	58.95	65.86
Dense_ctx0+ss.agglo	126	76.24	60.5	**77.61**	68
Combined	239	**77.03**	**65.23**	73.82	**69.26**
@Combined+sep. *sell*	240	78.86	70.61	73.82	72.18
⋆Dense_ctx7+ss.agglo	194	77.52	66.68	**72.67**	69.55
⋆Combined2	272	**78.15**	**70.86**	70.54	**70.70**
⋆Dense_ctx7+maxsil	126	75.77	60.23	76.34	67.33

Table 2: Subtask-A, results on test. ⋆ for post-eval results, @ for manual postprocessing (out of competition)

Pattern	ctx	PuIpuF1	B3P	B3R	B3F1	maxB3F1
T and then _	2	**78.15**	**70.86**	70.54	**70.70**	**71.34**
T and _	2	77.92	70.43	70.16	70.30	71.16
_	2	77.80	70.37	69.85	70.11	71.01
T	2	77.95	68.50	**71.97**	70.19	71.15
T and then _	0	77.79	70.56	69.67	70.11	71.06
T and then _	1	77.93	70.87	69.89	70.38	**71.38**
T and then _	2	**78.15**	70.86	70.54	**70.70**	71.34
T and then _	3	78.14	70.52	**70.66**	70.59	71.29
T and then _	5	77.72	70.29	70.10	70.19	71.13
T and then _	7	77.94	**70.95**	69.89	70.41	71.24

Table 3: Subtask-A, effect of pattern and context size

an appropriate pattern are better for disambiguating homonyms.

We denote the proportion of synonyms sharing common cluster as recall for synonyms and the proportion of homonyms put in separate clusters as recall for homonyms. Figure 1 shows both metrics depending on the number of clusters for agglomerative clustering of the whole development set. It is evident that until a relatively large number of clusters (30) almost all synonyms are correctly clustered together when using dense representations, yet homonyms are clustered together as well, which gives almost 1.0 recall for synonyms and nearly 0.0 recall for homonyms. MaxB3f1 of approximately 0.94 is achieved at around 25-28 clusters (depending on the context size) where synonyms are still clustered almost perfectly. At the same time, sparse representations split homonyms into different clusters even at very small numbers of clusters, but simultaneously split synonyms also, achieving lower maxB3f1 of 0.91 in a wider range of 25-40 clusters. To solve this problem, our final solution clusters dense representations first and then splits large clusters containing examples of the same verb (to prevent splitting synonyms) into a small number of clusters to improve recall for homonyms.

Table 2 compares results on the test set. Verb baseline assigns the first token of the verb to each example as its cluster id. It overestimates the real number of clusters in the test (149), giving the highest precision but very low recall because synonyms are never clustered together. Dense representation with semi-supervised agglomerative clustering slightly underestimates the number of clusters in the test set (similarly to the development set) resulting in the highest recall due to merged synonyms. The combined approach splits some clusters hurting BCubed-recall a bit but increasing BCubed-precision, even more, resulting in better BCubed-f1. The last row shows that selecting the number of clusters which maximizes silhouette score (unsupervised approach) instead of BCubed-f1 of the labeled subset results in much worse results, hence our semi-supervised approach is beneficial. Finally, we noticed that the largest cluster had all the examples of both *sell* and *buy*, which were among the most frequent verbs in the test set. In FrameNet, they are assigned to *Commerce_sell* and *Commerce_buy* frames respectively which is a questionable solution since these are just different ways to put into words the same type of event with the same participants (something like commercial-transfer-of-property). We simply moved all examples of the verb *sell* into a separate cluster which gave significant improvement in BCubed-f1. However, this result is out of competition due to the manual postprocessing. Yet, our best result without manual postprocessing is still ranked first.

In Table 3 we report the results of clustering the test set depending on the pattern and the context size used to build sparse representations at phase 2. In addition to standard metrics, we report maxB3F1 which excludes the effect of a suboptimal number of clusters selected on the comparison results. Our proposed pattern seems to give small but consistent improvement as well as context extension. The context of 1-3 sentences on both sides is a reasonable choice for sparse representations.

4 Semantic Role Induction

After looking at examples from the development set we decided that the subtask B.2 (generic semantic role induction) could be solved much more effectively using a classifier than any kind of clustering because generic roles look more like a high-level linguistic abstraction than something naturally occurring in texts. We used the development set to trained logistic regression on top of representations extracted from BERT and several hand-crafted features. BERT was pretrained in unsupervised fashion on large corpora and this results in much better generalization of our semi-supervised approach compared to a logistic regression trained only on hand-crafted features (see ablation analysis below). To select hyperparameters we used cross-validation with lexical split (i.e. there were no common verbs in train and test subsets for each fold) to ensure the best performance on new verbs not seen during training. This approach was rejected as using an additional labeled corpora to train a supervised component. However we hardly see how the development set provided by the organizers can be considered as additional.

4.1 Model Description and Results

We trained a logistic regression classifier for the 14 most frequent semantic roles in the development set. Following recommendations of Devlin et al. (2018) we used outputs from the last four layers of BERT as features. These outputs were taken for two timesteps at which the target argument and its corresponding verb were fed. To be exact, we found the first subword of the verb (for instance, *buy* for *buy out*) and the last subword for the argument (*Union* for *European Union*) performing best. Additionally we used several hand-designed features. Table 4 shows our submission results. Also, we display results when using only BERT and only hand-designed features suggesting that both of them contribute positively

Method	#cl	PuIpuF1	B3P	B3R	B3F1
ClstPerGrType	37	56.05	40.89	37.33	39.03
Logistic regression	14	**77.47**	**56.21**	**74.41**	**64.04**
w/o designed feats.	14	76.93	54.71	73.55	62.75
w/o BERT feats.	13	65.08	41.90	55.01	47.57

Table 4: Subtask-B.2, results on test

to the results but BERT features are much more important. For additional details regarding hand-designed features and ablation analysis please refer to Appendix B. We didn't experiment with subtask B.1 due to the lack of time, instead we used labels predicted for subtask B.2 which resulted in 64.43 / 73.11 BCubed-F1 / PuIpu-F1 compared to 45.79 / 57.99 of the best performing baseline.

5 Conclusions

We show how neural language models can be effectively used for unsupervised inference of semantic structures. To improve the result of semantic frame induction we used a combined approach that utilizes two different vector representations, and adjusted our clustering algorithm accordingly. The design stemmed from our analysis of problems in use of neural language models for the purpose of semantic frame induction; the experiments showed that issues may be strongly related to how the models treat such linguistic phenomena as synonymy and homonymy. Designing a system that addresses this problem directly allowed us to improve the result significantly. We think that our result could be additionally improved by finding better parameters and/or model combinations. We also think that further research in this direction could lead to neural language models that explicitly address various linguistic phenomena by design, for even better inference of semantic properties.

Acknowledgments

We thank Dmitry Lipin and Dmitry Ustalov for their invaluable help, the orginizers and LDC for the inspiring task and the data, all the reviewers for the useful feedback. Alexander Panchenko has been supported by the Deutsche Forschungsgemeinschaft (DFG) within the project "Argumentation in Comparative Question Answering (ACQuA)" (grant BI 1544/7-1 and HA 5851/2-1) that is part of the Priority Program "Robust Argumentation Machines (RATIO)" (SPP-1999) and the project "JOIN-T".

References

Asaf Amrami and Yoav Goldberg. 2018. Word sense induction with neural bilm and symmetric patterns. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4860–4867. Association for Computational Linguistics.

Nikolay Arefyev, Pavel Ermolaev, and Alexander Panchenko. 2018. How much does a word weigh? Weighting word embeddings for word sense induction. In *Computational Linguistics and Intellectual Technologies. Papers from the Annual International Conference Dialogue (2018)*, pages 68–84. RSUH.

Osman Baskaya, Enis Sert, Volkan Cirik, and Deniz Yuret. 2013. Ai-ku: Using substitute vectors and co-occurrence modeling for word sense induction and disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 300–306. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics*.

David Jurgens and Ioannis Klapaftis. 2013. Semeval-2013 task 13: Word sense induction for graded and non-graded senses. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 290–299. Association for Computational Linguistics.

Alexander Panchenko, Anastasiya Lopukhina, Dmitry Ustalov, Konstantin Lopukhin, Nikolay Arefyev, Alexey Leontyev, and Natalia V. Loukachevitch. 2018. Russe'2018: A shared task on word sense induction for the russian language. In *Computational Linguistics and Intellectual Technologies. Papers from the Annual International Conference Dialogue (2018)*, pages 547–564. RSUH.

Alexander Panchenko, Olga Morozova, and Hubert Naets. 2012. A semantic similarity measure based on lexico-syntactic patterns. In *KONVENS*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Behrang QasemiZadeh, Miriam R. L. Petruck, Regina Stodden, Laura Kallmeyer, and Marie Candito. 2019. Semeval-2019 task 2: Unsupervised lexical frame induction. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Roy Schwartz, Roi Reichart, and Ari Rappoport. 2015. Symmetric pattern based word embeddings for improved word similarity prediction. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 258–267. Association for Computational Linguistics.

Oleg Struyanskiy and Nikolay Arefyev. 2018. Neural Networks with Attention for Word Sense Induction. In *Supplementary Proceedings of the Seventh International Conference on Analysis of Images, Social Networks and Texts (AIST 2018)*, pages 208–213.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR*, abs/1706.03762.

Dominic Widdows and Beate Dorow. 2002. A graph model for unsupervised lexical acquisition. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

build: Manufacturing/ Building	**follow**: Compliance/ Relative_time	**join**: Participation/ Becoming_a_member
drive 0.61/0.031	execute 0.53/0.0	end 0.5/0.15
export 0.5/0.031	obey 0.47/0.0	support 0.81/0.32
import 0.67/0.046	keep 0.53/0.0	continue 0.5/0.23
distribute 0.72/0.092	adopt 0.74/0.0	begin 0.44/0.21
manufacture 0.94/0.12	apply 0.68/0.013	follow 0.56/0.37
ship 0.5/0.077	maintain 0.63/0.013	lead 0.88/0.77
release 0.5/0.092	use 0.63/0.013	start 0.5/0.56
make 0.56/0.11	enforce 0.47/0.013	leave 0.5/0.82
assemble 0.78/0.18	ignore 0.42/0.013	represent 0.31/0.52
deliver 0.89/0.22	implement 0.79/0.027	rejoin 0.25/0.6
...	...	...
rebuild 0.28/0.55	confirm 0.16/0.47	buy 0.062/0.47
expand 0.33/0.75	begin 0.16/0.48	found 0.0/0.4
acquire 0.22/0.58	end 0.11/0.64	oversee 0.0/0.4
finance 0.17/0.58	see 0.053/0.43	serve 0.0/0.48
erect 0.11/0.43	include 0.053/0.55	create 0.0/0.48
open 0.11/0.77	come 0.0/0.41	acquire 0.0/0.48
fund 0.056/0.58	be 0.0/0.49	purchase 0.0/0.55
establish 0.056/0.6	, 0.0/0.51	establish 0.0/0.6
close 0.0/0.42	mark 0.0/0.53	form 0.0/0.63
start 0.0/0.43	after 0.0/0.61	become 0.0/0.82

Table 5: Examples of generated substitutes for template "T and then _"

Method	#cl	PuIpuF1	B3P	B3R	B3F1
ClstPerGrType	37	56.05	40.89	37.33	39.03
Logistic regression	14	**77.47**	**56.21**	**74.41**	**64.04**
w/o designed feats.	14	76.93	54.71	73.55	62.75
w/o BERT feats.	13	65.08	41.90	55.01	47.57
w/o BERT@arg	14	73.45	51.60	67.59	58.52
w/o BERT@verb	14	74.88	52.16	71.88	60.45
w/o verb_input_rel	14	76.92	55.45	73.45	63.19
w/o by_between	14	76.92	55.55	73.45	63.25
w/o arg_is_left	14	77.25	55.65	73.90	63.49
layer 0,1	14	73.56	50.34	68.33	57.97
layer 0	14	73.69	50.52	68.88	58.29
layer 1	14	74.71	51.65	70.47	59.61
layer 2	14	75.13	52.18	71.16	60.21
layer 4	14	76.16	54.11	72.30	61.90
layer 11	14	75.98	54.37	72.19	62.02
layer 10,11	14	76.58	55.10	73.25	62.89
layer 10	14	76.66	55.51	72.96	63.05
layer 6	14	76.98	55.72	73.29	63.31
layer 8	14	77.40	56.33	73.78	63.88

Table 6: Subtask-B.2, ablations on test set.

A Examples of generated substitutes

To show how substitutes can disambiguate homonyms we generated substitutes for examples of two most frequent frames for several verbs. For each verb we excluded rare substitutes with $P(subs|frame_i) < 0.4$ for both frames. Then we sorted the rest according to the probability ratio $\frac{P(subs|frame_1)}{P(subs|frame_2)+1e-6}$. Table 5 shows substitutes with the largest and the smallest ratio (most discriminating substitutes).

B Features and ablation analysis for Generic Semantic Role Induction subtask

We used the following hand-crafted features: an indicator that the argument is to the left of the verb and an indicator that the particle *by* is between them; categorical features for the output syntactic relation of the argument, the last relation in the path between the argument and the verb, the part of speech of the first word of the argument, the number of words and the number of words starting with a capital letter in the argument. All these features were concatenated, categorical features were encoded with one-hot vectors. In the preliminary experiments we noticed that hand-crafted features performed well by themselves but didn't improve results when concatenated with BERT outputs; this was resolved by multiplying the features by 10 (we attribute the effect to very high dimensionality of BERT outputs compared to hand-crafted features, which requires harmonizing the variance each of them adds to the scalar product in the logistic regression). We tried multiplying each feature by its own constant determined analytically from its dimensionality, but this worsened the results, so we left it for the future work.

Table 6 shows results for subtask B.2 after removing features from input representation or using different BERT layers instead of the last four. For ablation analysis, we selected L2-regularization strength using cross-validation with a lexical split after removing each feature while leaving all other hyperparameters intact. The features with largest contribution to the result are (from most to least important) BERT output at the argument, at the verb, the last relation in the path from the argument to the verb, the indicator that the particle *by* is between them (which was designed to fix errors due to passive voice) and the indicator that the argument is to the left of the verb. All other features' contributions (not shown) are small. Remarkably, removing all BERT features gives very large decrease in performance (-18 B3F1) while removing only outputs at the argument/verb gives only moderate decrease (-5.5/-3.5 B3F1) which can be explained by deeply bidirectional nature of BERT resulting in some information about both the verb and the argument present in each of these outputs. Finally, we tried using other BERT layers instead of the last four (layers 8-11) and found that intermediate layers perform best. For instance, layer 8 can replace the last four layers with very little decrease in performance, while the last two layers (10, 11) concatenated perform noticeably worse but much better than the first layers.

SemEval-2019 Task 3: EmoContext Contextual Emotion Detection in Text

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi and Puneet Agrawal
Microsoft, India
{anchatte, kedharn, mejoshi, punagr}@microsoft.com

Abstract

In this paper, we present the SemEval-2019 Task 3 - EmoContext: Contextual Emotion Detection in Text. Lack of facial expressions and voice modulations make detecting emotions in text a challenging problem. For instance, as humans, on reading "*Why don't you ever text me!*" we can either interpret it as a sad or angry emotion and the same ambiguity exists for machines. However, the context of dialogue can prove helpful in detection of the emotion. In this task, given a textual dialogue i.e. an utterance along with two previous turns of context, the goal was to infer the underlying emotion of the utterance by choosing from four emotion classes - *Happy*, *Sad*, *Angry* and *Others*. To facilitate the participation in this task, textual dialogues from user interaction with a conversational agent were taken and annotated for emotion classes after several data processing steps. A training data set of 30160 dialogues, and two evaluation data sets, Test1 and Test2, containing 2755 and 5509 dialogues respectively were released to the participants. A total of 311 teams made submissions to this task. The final leader-board was evaluated on Test2 data set, and the highest ranked submission achieved 79.59 micro-averaged F1 score. Our analysis of systems submitted to the task indicate that Bi-directional LSTM was the most common choice of neural architecture used, and most of the systems had the best performance for the *Sad* emotion class, and the worst for the *Happy* emotion class.

1 Introduction

Emotions are basic human traits and have been studied by researchers in the fields of psychology, sociology, medicine, computer science etc. for several years. Some of the prominent work in understanding and categorizing emotions include Ekman's six class categorization (Ekman, 1992) and Plutchik's "Wheel of Emotion" (Plutchik and Kellerman, 1986) which suggested eight primary bipolar emotions . In recent times, several Artificial Intelligence (AI) agents like Siri, Cortana, Alexa have emerged and they primarily focus on providing users with assistance on specific tasks such as booking tickets or scheduling meetings etc. However, we believe that for machines and humans to develop a deeper partnership, an Intelligence Quotient (IQ) is not enough. These agents need to also possess an Emotional Quotient (EQ). Social conversational agents like Mitsuku[1] or Ruuh [2] (Damani et al., 2018) are experimental agents designed to have human-like persona, and possess a deeper sense of EQ; understanding and expressing emotions is an inherent aspect of these agents.
Detecting emotions in textual dialogues is a challenging problem in absence of facial expressions and voice modulations. Moreover, we observed that context of ongoing dialogue can completely change the emotion for an utterance as compared to perceived emotion when the utterance is evaluated standalone. Table 1 presents few such examples. Note that, in the first example *"I started crying"* will be perceived as '*Sad*' by a majority, however considering it in context, it turns out to be a '*Happy*' emotion. Similarly, in the second example, the last turn *"Try to do that once"* is very likely to be perceived as '*Others*', however again, a majority will judge it as '*Angry*' with the given context.

Naturally, considering context to estimate emotion of a text utterance becomes even more important for aforementioned scenarios of digital assistants and conversational agents, because of their text-based conversational interface. This task was

[1]www.pandorabots.com/mitsuku
[2]www.ruuh.ai

User Turn-1	Conversational Agent Turn-1	User Turn-2	True Class
I just qualified for the Nabard internship	WOOT! Thats great news. Congratulations!	I started crying	Happy
How dare you to slap my child	If you spoil my car, I will do that to you too	Just try to do that once	Angry
I was hurt by u more	You didn't mean it.	say u love me	Sad

Table 1: Examples showing influence of context in determining emotion of last utterance.

designed to invite research interest in the area of emotion detection in text. More details about the task can be found on our web page[3]. The evaluation data set served as a benchmark to compare various techniques and the task received attention from a wide range of researchers from industry as well as academia. We believe continued interest in this field will be beneficial towards making the AI-agents more human-like.

2 Related Work

Researchers have achieved good results on image based emotion recognition (Wang et al., 2018), (Zhang et al., 2016) as well as voice based emotion recognition (Pierre-Yves, 2003). Techniques have been proposed to detect emotions in spoken dialog systems (Liscombe et al., 2005). However, classifying textual dialogues based on emotions is relatively new research area. Emotion-detection algorithms for text can be largely bucketized into following two categories:

(a) *Hand-crafted Feature Engineering Based Approaches:* - Many methods exploit the usage of keywords in a sentence with explicit emotional/affect value (Balahur et al., 2011), (Strapparava and Mihalcea, 2008), (Sykora et al., 2013). To that end, several lexical resources have been created, such as WordNet-Affect (Strapparava et al., 2004) and SentiWordNet (Esuli and Sebastiani, 2007). Part-of-Speech taggers like the Stanford POS tagger are also used to exploit the structure of keywords in a sentence. These pattern/dictionary based approaches, although attaining high precision scores, suffer from low recall.

Hasan et al. (2014), Purver and Battersby (2012), Suttles and Ide (2013) and Wang et al. (2012) have also harnessed cues from emoticons and hashtags. Other methods rely on extracting statistical features such as presence of frequent n-grams, negation, punctuation, emoticons, hashtags to form representations of sentences which are then used as input by classifiers such as Decision Trees, SVMs among others to predict the output (Alm et al., 2005), (Balabantaray et al., 2012), (Davidov et al., 2010), (Kunneman et al., 2014), (Yan and Turtle, 2016). However, all of these methods require extensive feature engineering and they often do not achieve high recall due to diverse ways of representing emotions. For example, the following utterance, *"Trust me! I am never gonna order again"*, contains no affective words despite conveying an emotion of anger or frustration perhaps.

(b) *Deep Learning Based Approaches:* - Deep Neural networks have enjoyed considerable success in varied tasks in text, speech and image domains. Variations of Recurrent Neural Networks, such as Long Short Term Memory networks (LSTM) (Hochreiter and Schmidhuber, 1997) and Bidirectional LSTM (BiLSTM) (Schuster and Paliwal, 1997) have been effective in modeling sequential information. Also, Convolutional Neural Networks (CNN) (Krizhevsky et al., 2012) have been a popular choice in the image domain. Their introduction to the text domain has proven their ability to decipher abstract concepts from raw signals (Kim, 2014).
Recently, approaches which employ Deep Learning for emotion detection in text have been proposed. Zahiri and Choi (2017) predicts emotion in a TV show transcript. Abdul-Mageed and Ungar (2017) and Köper et al. (2017) tries to understand emotions of tweets. Li et al. (2017) learns to detect emotions on user comments in Chinese language. Felbo et al. (2017) learns representation based on emoticons, and uses it for emotion detection. A further detailed analysis of various approaches have been provided by Chatterjee et al. (2019). It is worth noting that textual dialogues are informal and laden with misspellings which pose serious challenges for automatic emotion detection approaches. Prior to this task, to the best of our knowledge, the methods proposed by Mundra

[3]Task webpage: humanizing-ai.com/emocontext.html

et al. (2017) and Chatterjee et al. (2019) are some of the few methods that tackled the problem of emotion detection in English textual dialogues.

3 Task Details

Problem Definition: *In a textual dialogue, given an utterance along with its two previous turns of context, classify the emotion of the utterance as one of the following classes: Happy, Sad, Angry or Others.*

The motivation for restricting the number of emotion classes stems from the popularity of these emotions in conversational data. The task proceeded in two phases. A training corpus, Train, of 30160 dialogues was provided at the beginning of Phase 1. The evaluation in this phase was done on an evaluation data set, Test1, comprising of 2755 dialogues. The labels for Test1 were made public five weeks before the end of Phase 1, allowing participants time and data to improve their models. The final evaluation was carried out in Phase 2 on a evaluation data set, Test2, which comprised of 5509 dialogues. It is important to note that while the maximum number of submissions a participant could make in Phase 1 was 20 per day, it was reduced to 10 per day during Phase 2.

4 Data Collection

A data set of textual dialogues was released to facilitate participation in this task. Several data processing steps were performed to create the final set of textual dialogues which are further explained in this section.

4.1 Dialogue Collection and Processing

A dialogue mined from the user's interaction with agent is defined as a tuple of 3 values - User Turn-1 (Utterance of the user), Conversational Agent Turn-1 (Response by the agent), User Turn-2 (User utterance as response to agent).
To begin with, user interactions with the agent over a period of one year were considered and over 2 million dialogues were randomly sampled. These dialogues further went through the processing and data cleaning as described in further subsections.

4.1.1 Offensive filtering

All the dialogues were passed through a filtering layer to remove offensive and sensitive content such as adult information, politically sensitive topics, or ethnic-religious content, or other potentially contentious material, such as inappropriate references to violence, crime and illegal substances etc. Several lexicons and human judgments were used to achieve this filtering.

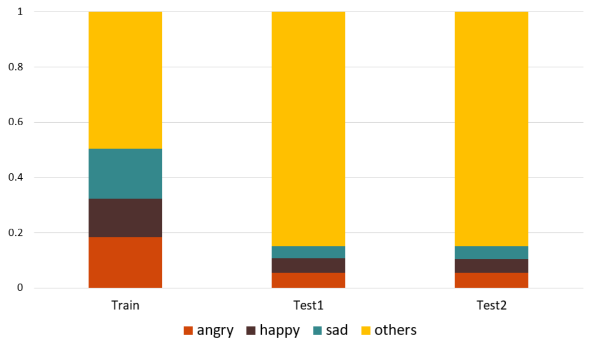


Figure 1: Comparison of class distribution in Training vs Evaluation data sets.

Emotion	Happy	Sad	Angry	Others	#
Train	4243	5463	5506	14948	30160
Test1	142	125	150	2338	2755
Test2	284	250	298	4677	5509

Table 2: Emotion label count across classes in Train, Test1 and Test2 data sets.

4.1.2 PII filtering

Personally Identifiable Information (PII) identifies the unique identity of a given user. This includes personal data like names, phone numbers, email Ids, among others. Dialogues containing any PII content were removed using hand crafted rules and via human judgments.

4.1.3 Language filtering

Given that the agent was available for users across geographies, the dialogues contained multiple languages and users employed code-mixed language as well. We used language detectors as well as user modeling to identify the language in the dialogues and filter non-English dialogues from the data set.

4.2 Training Data Set Creation

In the collected textual dialogues the emotion classes were not frequently expressed and hence directly annotating a random sample of textual dialogues results in very low volume of textual dialogues with emotion class. This problem was tackled by Gupta et al. (2017) and we used similar heuristics and strategies to ensure a higher ratio of

textual dialogues with emotion classes. This exercise was primarily conducted to reduce the cost of human judgments and is further explained below. We started with a small set (approximately 300) of annotated dialogues per emotion class obtained by showing a randomly selected sample to human judges. Using a variation of the model described by Palangi et al. (2016), we created embedding for these annotated dialogues. Potentially similar dialogues were further identified from the entire pool of dialogues using a threshold-based cosine similarity and these dialogues form our candidate set for each emotion class. Various heuristics like presence of opposite emoticons (example ":'(" in a potential candidate set for *Happy* emotion class), sentiment analysis, length of utterances etc. are used to further prune the candidate set in certain cases. The candidate set is then shown to human judges to determine if they belong to an emotion class. Using this method, we cut down the amount of human judgments required by five times as compared to showing a random sample of dialogues and then choosing dialogues with emotion class from them.
Data belonging to class "*Others*" is collected by randomly selecting dialogues from our pool of dialogues and were human labelled to discard any dialogues with emotion class such as *Happy*, *Sad* or *Angry*.
Figure 1 shows the distribution of different classes in training data set.

4.3 Evaluation Data Set Creation

Unlike training data set where we intentionally over sampled dialogues from emotion classes to help participants with a larger volume of data with emotion classes, we maintained the natural distribution of emotion classes in evaluation data sets. We randomly sampled and annotated two evaluation sets, Test1 and Test2, of size 2755 and 5509 respectively. Detailed distribution of emotion classes in these sets is described in Table 2.

4.4 Emotion Class Labeling

For this specific task of emotion class labelling, 50 human judges were trained. Given a dialogue, i.e an utterance with two previous turns as context, a judge was asked to annotate the utterance as belonging to one of the following four classes: *Happy*, *Angry*, *Sad* or *Others*. All dialogues were judged by 7 human judges and a majority consensus was taken as the final class label. Fleiss' Kappa score (Shrout and Fleiss, 1979) of 0.58 was observed on training data set and of 0.59 on evaluation data set. Such a Kappa score indicates the existence of multiple perspectives about the underlying emotion of a conversation.

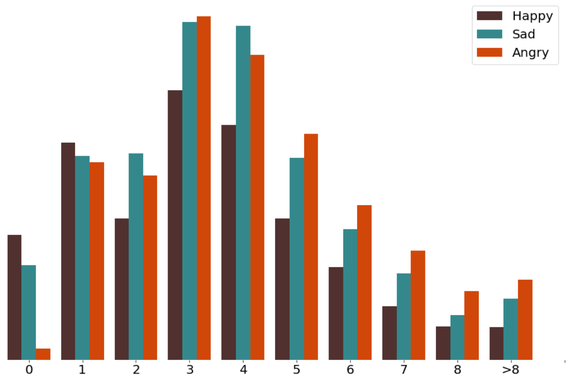


Figure 2: Comparison of word count of utterances per emotion class. Emoticons were removed for this calculation, as a result of which the leftmost bin of 0 word count can be seen as well.

5 Data Analysis

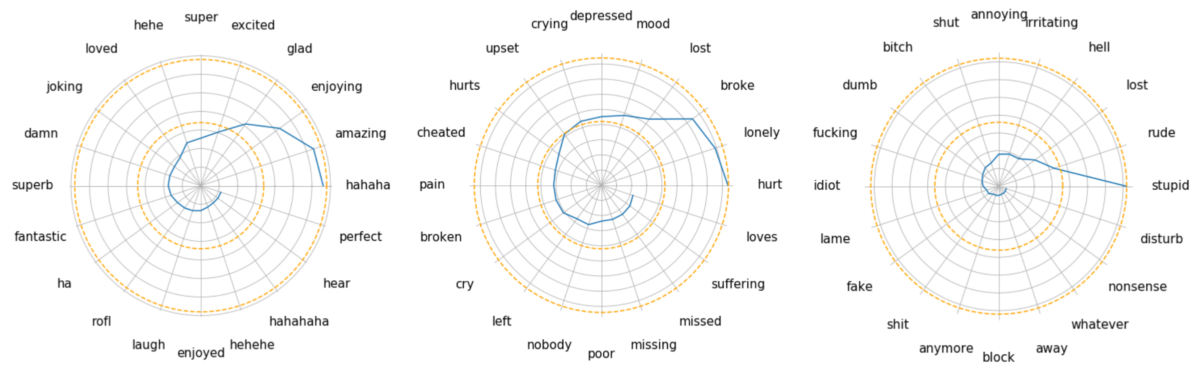
In this section we analyze the utterance in the dialogue that was judged by human judges for emotion classes.

5.1 Word Count

Figure 2 shows the distribution of the word count of utterances per emotion class. We observed that users tend to repeat emoticons several times. Hence emoticons were removed from utterances for this calculation, as a result of which the utterances which had only emoticons are clubbed in the leftmost bin with utterance of length 0. It can be observed that happiness is often expressed through emoticons and hence happy emotion class has highest count under the bin of 0 word count. Also, happiness is often expressed in fewer words as compared to other emotions can be observed from the graph. Another point to note is that angry emotion class is often expressed using more words as compared to other emotion classes.

5.2 Top Unigrams

Figure 3 shows the most frequent unigrams per emotion class in our data set. Note that emoticons are not considered as unigrams for this analysis. The length of the radius in the spiral graph denotes the frequency of the unigram in all the utterances belonging to that particular emotion class. In order



(a) Happy (b) Sad (c) Angry

Figure 3: Most frequent unigrams per emotion class in our data set. The length of the radius in the spiral graph denotes the frequency of the unigram in all the utterances for a emotion class. Only those unigrams which are not in the top 500 list of most frequent unigrams of the “Others” class have been considered.

Happy	😂	😁	😀	😄	😄
Sad	😭	😞	😢	💔	😔
Angry	😡	😠	😒	😑	😤

Table 3: Top five emoticons per emotion class.

to avoid neutral words like “my”, “what”, “sure” from showing up in the analysis, we consider only those unigrams which are not in the top 500 list of most frequent unigrams of the “Others” class.

5.3 Top Emoticons

Emoticons are frequently used in textual dialogues, as was observed by Gupta et al. (2017), who found 21% of textual dialogues to contain emoticons. Table 3 shows the top emoticons observed in utterances per emotion class. While most emoticons align with our expectations of the most frequent emoticons, it is interesting to note the frequent use of broken-heart emoticon to express sad emotion.

6 Evaluation Metric

Evaluation was carried out using the micro-averaged F1 score ($F1_\mu$) for the three emotion classes - *Happy*, *Sad* and *Angry* on the submissions made with predicted class of each sample in the evaluation data set. To be precise, we define the metric as following:

$$P_\mu = \frac{\Sigma TP_i}{\Sigma(TP_i + FP_i)} \forall i \epsilon \{Happy, Sad, Angry\}$$

$$R_\mu = \frac{\Sigma TP_i}{\Sigma(TP_i + FN_i)} \forall i \epsilon \{Happy, Sad, Angry\}$$

$$F1_\mu = 2 \cdot \frac{P_\mu \cdot R_\mu}{P_\mu + R_\mu}$$

where TP_i is the number of samples of class i which are correctly predicted, FN_i and FP_i are the counts of *Type-I and Type-II errors* [4] respectively for the samples of class i.

Our final metric $F1_\mu$ is calculated as the harmonic mean of P_μ and R_μ.

7 Baseline Model

To encourage and assist participants in making their first submission, we provided a starter kit, which consisted of scripts for training a naive baseline model. The script also enabled participants to cross-validate their model and create a submission file. This section explains the baseline model in detail.

7.1 Data Processing

Minimal data pre-processing steps were provided. These included replacing certain repeated punctuation marks with their single instances, lower casing, removing extra space and tokenization. For example, “I am so happy!!” was converted to “i am so happy !”.

7.2 Model Architecture

We modeled the task of detecting emotions as a multi-class classification problem where given a dialogue, the model outputs probabilities of it belonging to four output classes - *Happy*, *Sad*, *Angry* and *Others*. The three turns are concatenated using a special <*eos*> token. The concatenated input is passed into a pre-trained word embedding

[4] http://en.wikipedia.org/wiki/Type_I_and_type_II_errors

Team	GloVe	Word2Vec	NTUA-SLP	BERT	ELMO	ULMFit	Others
NELEC							✓
SymantoResearch	✓			✓			✓
ANA	✓			✓	✓		
CAiRE_HKUST	✓			✓	✓		✓
SNU_IDS		✓			✓		✓
THU-HCSI		✓	✓				
Figure Eight			✓	✓		✓	✓
YUN-HPCC	✓				✓		
LIRMM-Advanse						✓	
MILAB	✓						
PKUSE	✓						
THU_NGN	✓	✓					✓

Table 4: Input representations used by top systems.

layer, which projects the words into continuous vector representations. We used 100 dimensional GloVe embeddings (Pennington et al., 2014) for this purpose. The embeddings are processed by an LSTM layer, which produces a 128 dimensional representation of the sentence. This representation is then mapped to a 4 dimensional output vector which outputs probabilities per emotion class using a fully connected neural network. The architecture of the model was kept deliberately simple and was intended to serve as a starting point for participants. The baseline model achieved a $F1_\mu$ score of 0.5861 on the final leader board and most teams were able to beat the baseline model. Further details on the model and its comparison with other systems can be seen in Table 5.

8 Systems and Results

As mentioned earlier in section 3, the task was conducted in two phases. The first phase saw a participation from 311 teams and 164 teams participated in the second phase. In this section, we briefly describe the top systems [5], followed by observations across systems regarding the techniques used and their performance across different emotion classes.

[5]The top 2 systems - *Leo1020* and *Mfzszgs* did not submit system description papers, and hence have been omitted from discussion in this Section.

8.1 Top Systems

Due to the overwhelming number of participants, we cannot describe all systems. We describe the main features of the top few systems ranked according to their final performance.

- **NELEC** uses a combination of lexical features such as word and character grams, along with additional signals like emotional intensity, valence-arousal-dominance scores. In addition, they use adult, offensive and sentiment classifiers' scores from neural models. Using these features, the authors trained a Light-GBM tree (Ke et al., 2017), which achieves better performance than their deep-learning based architecture.

- **SymantoResearch** explores different deep-learning based architectures, some of them employing multi-task learning to better classify *Others* class vs. emotion classes. By ensembling such architectures with fine-tuned BERT (Devlin et al., 2018) and USE (Cer et al., 2018) models, the authors are able to distinguish three emotions (*Sad*, *Happy*, *Angry*) and separate them from the rest (*Others*) more accurately.

- **ANA** uses an ensemble of fine tuned BERT model and Hierarchical LSTMs, where the semantic and emotional content of text is encoded via GloVe, ELMo (Peters et al., 2018)

Team Name	Angry			Happy			Sad			F1$_\mu$
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	
Leo1020	0.7723	0.8423	**0.8058**	**0.804**	0.7077	**0.7528**	0.8494	0.812	0.8303	**0.7959**
Mfzszgs	0.759	0.8456	0.8	0.7769	0.7113	0.7426	**0.8595**	0.832	**0.8455**	0.7947
NELEC	0.747	0.8322	0.7873	0.7632	0.7148	0.7382	0.7938	0.816	0.8047	0.7765
SymantoResearch	**0.7807**	0.7886	0.7846	0.738	0.7042	0.7207	0.8193	0.816	0.8176	0.7731
ANA	0.7198	0.8188	0.7661	0.7698	0.6831	0.7239	0.8458	0.812	0.8286	0.7709
CAiRE_HKUST	0.6997	0.8289	0.7588	0.7301	**0.743**	0.7365	0.7774	**0.852**	0.813	0.7677
SNUIDS	0.7405	0.7852	0.7622	0.772	0.6796	0.7228	0.8135	0.82	0.8167	0.7661
THU-HCSI	0.7155	0.8356	0.7709	0.7702	0.6725	0.718	0.796	0.796	0.796	0.7616
Figure Eight	0.6954	**0.8658**	0.7713	0.7055	0.7254	0.7153	0.7695	0.828	0.7977	0.7608
YUN-HPCC	0.7198	0.8188	0.7661	0.7169	0.6866	0.7014	0.8016	0.824	0.8126	0.7588
LIRMM-Advanse	0.7229	0.8054	0.7619	0.7256	0.7077	0.7166	0.8291	0.776	0.8017	0.7582
MILAB	0.7295	0.8054	0.7656	0.7481	0.7007	0.7236	0.7652	0.808	0.786	0.7581
Huxiao	0.7362	0.8054	0.7692	0.7403	0.6725	0.7048	0.7757	0.816	0.7953	0.7564
PKUSE	0.745	0.755	0.75	0.7351	0.6937	0.7138	0.8056	0.812	0.8088	0.7557
THU_NGN	0.7329	0.7919	0.7613	0.7452	0.6796	0.7109	0.8117	0.776	0.7935	0.7542
Baseline	0.4777	0.7867	0.5945	0.5123	0.5845	0.5461	0.5163	0.7600	0.6149	0.5861

Table 5: Performance comparison of top 15 teams on leaderboard.

and DeepMoji (Felbo et al., 2017) embeddings, following which a contextual LSTM encodes the entire dialogue for prediction.

- **CAiRE_HKUST** experiments with combinations of feature based models and end-to-end neural models. The feature based models use various pre-trained word embeddings and emotional embeddings, combining them with Logistic Regression and XGBoost (Chen and Guestrin, 2016). For the end-to-end neural models, the authors found the performance of hierarchical models, which take sequential nature of dialogue into account, to be better.

- **SNU_IDS** proposes several methods for alleviating the problems caused by difference in class distributions between training data and test data. The authors also present a semi-hierarchical neural architecture combining character and word embeddings that effectively encodes an utterance in context of the previous utterances.

- **THU-HCSI** is composed of three CNN-based neural network models trained for different base tasks - four-emotion classification, *Angry-Happy-Sad* classification and *Others-or-not* classification respectively. The authors use multiple steps of voting to combine the predictions of these base classifiers, resulting in a more accurate and robust model performance.

- **Figure Eight** uses an ensemble of transfer learning models for capturing the representations of the utterances. Using sophisticated fine-tuning techniques described in ULMFiT (Howard and Ruder, 2018), the authors observe that transfer learning using pre-trained language models outperforms models trained from scratch.

8.2 Miscellaneous Observations

From the system description papers of the top 15 teams, we observed that BiLSTMs/LSTMs were the most frequently used neural models. GRU (Chung et al., 2014) and CNN models were used by a few teams, and some variations of attention mechanism were employed by most of the teams to enhance performance of their models. Transfer learning using BERT, ELMo, ULMFit was a popular choice among top teams, and almost all the teams used an ensemble of their best models to create the final model.

	F1$_\mu$
Max	0.7959
Min	0.0143
Mean	0.6599
Median	0.694
1st Quartile	0.637
3rd Quartile	0.7317
Std. Dev.	0.1264

Table 6: Performance statistics of all participants.

Table 4 shows the embeddings used by the top 5 teams. It can be observed that GloVe was used most frequently. BERT and ELMo were the most popular choice for transfer learning. NTUA-SLP embeddings (Baziotis et al., 2018) were used as well to leverage its affective information. Participant teams tried various ways to encode the emotional content expressed by emoticons, and Deepmoji and Emoji2Vec (Eisner et al., 2016) were utilized in this regard. A good number of teams used the "ekphrasis" package (Baziotis et al., 2017) for tokenization, word normalization and word segmentation.

8.3 Performance across Emotion Classes

Table 5 displays the detailed performance of the top 15[6] participant teams. Upon inspection, it can be observed that the performance of the systems on the *Happy* class was not as good as the other emotion classes for the evaluation set. We believe, this is largely due to the natural ambiguity existing between neutral and happy utterances. For example, a greeting like "*Happy Morning*" can be thought of as expressing a happy emotion by some, while being judged to be neutral by others. We also observed that most systems performed best for the *Sad* emotion class. Table 6 provides some basic statistics on the results obtained by the whole set of participants.

9 Conclusion

A total of 311 teams made submissions to the task. The final leader-board was evaluated on Test2 data set, and the highest ranked submission achieved 79.59 $F1_\mu$ score. Our analysis of systems submitted to the task indicate that Bi-directional LSTM was the most common choice of network architecture used by participants, and most systems had best performance for *Sad* emotion class, and worst for *Happy* emotion class. A large number of teams have participated in the task but only 46 teams submitted their final system description papers; in fact, the top 2 teams in Phase 2 did not submit their system description paper. It was also observed that the ranking of various systems across both the phases varied significantly. In this task, we released the evaluation set without labels to participants, in future tasks it might be useful to also experiment with system submissions such that the entire evaluation set is never seen, with or without labels to the participants during the evaluation phase in a bid to have completely blind evaluation.

[6]Final rankings of all participating systems can be consulted via the CodaLab website of our task: https://competitions.codalab.org/competitions/19790

References

Muhammad Abdul-Mageed and Lyle Ungar. 2017. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume Vol. 1, pages 718–728.

Cecilia Ovesdotter Alm, Dan Roth, and Richard Sproat. 2005. Emotions from text: machine learning for text-based emotion prediction. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 579–586. ACL.

Rakesh C Balabantaray, Mudasir Mohammad, and Nibha Sharma. 2012. Multi-class twitter emotion classification: A new approach. *International Journal of Applied Information Systems*, Vol. 4, pages 48–53.

Alexandra Balahur, Jesús M Hermida, and Andrés Montoyo. 2011. Detecting implicit expressions of sentiment in text based on commonsense knowledge. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, pages 53–60. ACL.

Christos Baziotis, Nikos Athanasiou, Alexandra Chronopoulou, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, Shrikanth Narayanan, and Alexandros Potamianos. 2018. Ntua-slp at semeval-2018 task 1: predicting affective content in tweets with deep attentive rnns and transfer learning. *arXiv preprint arXiv:1804.06658*.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of* *the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. 2019. Understanding emotions in text using deep learning and big data. *Computers in Human Behavior*, 93:309–317.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Sonam Damani, Nitya Raviprakash, Umang Gupta, Ankush Chatterjee, Meghana Joshi, Khyatti Gupta, Kedhar Nath Narahari, Puneet Agrawal, Manoj Kumar Chinnakotla, Sneha Magapu, and Abhishek Mathur. 2018. Ruuh: A deep learning based conversational social agent. *32nd Conference on Neural Information Processing Systems (NIPS 2018), Montral, Canada*.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd international conference on computational linguistics: posters*, pages 241–249. ACL.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. *arXiv preprint arXiv:1609.08359*.

Paul Ekman. 1992. An argument for basic emotions. *Cognition & emotion*, Vol. 6, pages 169–200.

Andrea Esuli and Fabrizio Sebastiani. 2007. Sentiwordnet: A high-coverage lexical resource for opinion mining. *Evaluation*, pages 1–26.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524*.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations. *arXiv preprint arXiv:1707.06996*.

Maryam Hasan, Emmanuel Agu, and Elke Rundensteiner. 2014. Using hashtags as labels for supervised learning of emotions in twitter messages. In *ACM SIGKDD Workshop on Health Informatics, New York, USA*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, Vol. 9, pages 1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Maximilian Köper, Evgeny Kim, and Roman Klinger. 2017. Ims at emoint-2017: emotion intensity prediction with affective norms, automatically extended resources and deep learning. In *WASSA*, pages 50–57.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Florian Kunneman, Christine Liebrecht, and Antal van den Bosch. 2014. The (un) predictability of emotional hashtags in twitter. *In European Chapter of the Association for Computational Linguistics*, pages 26–34.

Panpan Li, Jun Li, Feiqiang Sun, and Peng Wang. 2017. Short text emotion analysis based on recurrent neural network. In *Proceedings of the 6th International Conference on Information Engineering*. ACM.

Jackson Liscombe, Giuseppe Riccardi, and Dilek Hakkani-Tur. 2005. Using context to improve emotion detection in spoken dialog systems.

Shreshtha Mundra, Anirban Sen, Manjira Sinha, Sandya Mannarswamy, Sandipan Dandapat, and Shourya Roy. 2017. Fine-grained emotion detection in contact center chat utterances. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 337–349. Springer.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, Vol. 24, pages 694–707.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume Vol. 14, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Oudeyer Pierre-Yves. 2003. The production and recognition of emotions in speech: features and algorithms. *International Journal of Human-Computer Studies*, 59(1-2):157–183.

Robert Plutchik and Henry Kellerman. 1986. *Emotion: theory, research and experience*. Academic press New York.

Matthew Purver and Stuart Battersby. 2012. Experimenting with distant supervision for emotion classification. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 482–491. ACL.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, Vol. 45, pages 2673–2681.

Patrick E Shrout and Joseph L Fleiss. 1979. Intraclass correlations: uses in assessing rater reliability. *Psychological bulletin*, Vol. 86, page 420.

Carlo Strapparava and Rada Mihalcea. 2008. Learning to identify emotions in text. In *2008 ACM symposium on Applied computing*, pages 1556–1560.

Carlo Strapparava, Alessandro Valitutti, et al. 2004. Wordnet affect: an affective extension of wordnet. In *The 4th International Conference on Language Resources and Evaluation*, volume Vol. 4, pages 1083–1086.

Jared Suttles and Nancy Ide. 2013. Distant supervision for emotion classification with discrete binary values. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 121–136. Springer.

Martin D Sykora, Thomas Jackson, Ann O'Brien, and Suzanne Elayan. 2013. Emotive ontology: Extracting fine-grained emotions from terse, informal messages. *IADIS International Journal on Computer Science and Information Systems*.

Shui-Hua Wang, Preetha Phillips, Zheng-Chao Dong, and Yu-Dong Zhang. 2018. Intelligent facial emotion recognition based on stationary wavelet entropy and jaya algorithm. *Neurocomputing*, 272:668–676.

Wenbo Wang, Lu Chen, Krishnaprasad Thirunarayan, and Amit P Sheth. 2012. Harnessing twitter "big data" for automatic emotion identification. In *Privacy, Security, Risk and Trust, 2012 International Conference on Social Computing*, pages 587–592. IEEE.

Jasy Liew Suet Yan and Howard R Turtle. 2016. Exploring fine-grained emotion detection in tweets. In *The North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 73–80.

Sayyed M Zahiri and Jinho D Choi. 2017. Emotion detection on tv show transcripts with sequence-based convolutional neural networks. *arXiv preprint arXiv:1708.04299*.

Yu-Dong Zhang, Zhang-Jing Yang, Hui-Min Lu, Xing-Xing Zhou, Preetha Phillips, Qing-Ming Liu, and Shui-Hua Wang. 2016. Facial emotion recognition based on biorthogonal wavelet entropy, fuzzy support vector machine, and stratified cross validation. *IEEE Access*, 4:8375–8385.

ANA at SemEval-2019 Task 3: Contextual Emotion detection in Conversations through hierarchical LSTMs and BERT

Chenyang Huang, Amine Trabelsi, Osmar R. Zaïane
Department of Computing Science, University of Alberta
{chuang8,atrabels,zaiane}@ualberta.ca

Abstract

This paper describes the system submitted by ANA Team for the SemEval-2019 Task 3: EmoContext. We propose a novel Hierarchical LSTMs for Contextual Emotion Detection (HRLCE) model. It classifies the emotion of an utterance given its conversational context. The results show that, in this task, our HRCLE outperforms the most recent state-of-the-art text classification framework: BERT. We combine the results generated by BERT and HRCLE to achieve an overall score of 0.7709 which ranked 5th on the final leader board of the competition among 165 Teams.

1 Introduction

Social media has been a fertile environment for the expression of opinion and emotions via text. The manifestation of this expression differs from traditional or conventional opinion communication in text (e.g., essays). It is usually short (e.g. Twitter), containing new forms of constructs, including emojis, hashtags or slang words, etc. This constitutes a new challenge for the NLP community. Most of the studies in the literature focused on the detection of sentiments (i.e. positive, negative or neutral) (Mohammad and Turney, 2013).

Recently, emotion classification from social media text started receiving more attention (Yaddolahi et al., 2017; Mohammad et al., 2018). Emotions have been extensively studied in psychology (Ekman, 1992; Plutchik, 2001). Their automatic detection may reveal important information in social online environments, like online customer service. In such cases, a user is conversing with an automatic chatbot. Empowering the chatbot with the ability to detect the user's emotion is a step forward towards the construction of an emotionally intelligence agent. Giving the detected emotion, an emotionally intelligent agent would generate an empathetic response. Although its potential convenience, detecting emotion in textual conversation has seen limited attention so far. One of the main challenges is that one users utterance may be insufficient to recognize the emotion (Huang et al., 2018). The need to consider the context of the conversion is essential in this case, even for human, specifically given the lack of voice modulation and facial expressions. The usage of figurative language, like sarcasm, and the class size's imbalance adds up to this problematic (Chatterjee et al., 2019a).

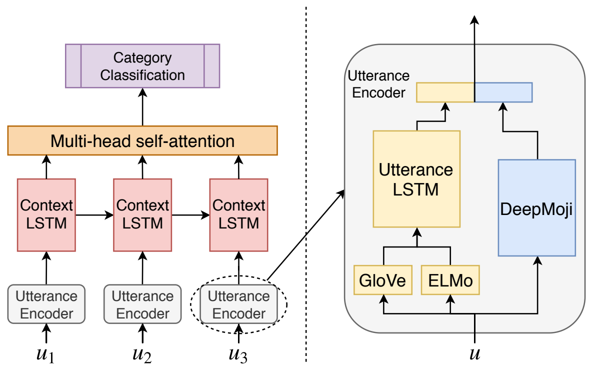


Figure 1: An illustration of the HRLCE model

In this paper, we describe our model, which was proposed for the SemEval 2019-Task 3 competition: Contextual Emotion Detection in Text (EmoContext). The competition consists in classifying the emotion of an utterance given its conversational context. More formally, given a textual user utterance along with 2 turns of context in a conversation, the task is to classify the emotion of user utterance as Happy, Sad, Angry or Others (Chatterjee et al., 2019b). The conversations are extracted from Twitter.

We propose an ensemble approach composed of two deep learning models, the *Hierarchical LSTMs for Contextual Emotion Detection* (HRLCE) model and the BERT model (Devlin et al., 2018). The BERT is a pre-trained language model that has shown great success in many NLP classification tasks. Our main contribution consists in devising the HRLCE model.

Figure 1 illustrates the main components of the HRLCE model. We examine a transfer learning approach with several pre-trained models in order to encode each user utterance semantically and emotionally at the word-level. The proposed model uses Hierarchical LSTMs (Sordoni et al., 2015) followed by a multi-head self attention mechanism (Vaswani et al., 2017) for a contextual encoding at the utterances level.

The model evaluation on the competition's test set resulted in a 0.7709 harmonic mean of the macro-F1 scores across the categories *Happy*, *Angry*, and *Sad*. This result ranked 5th in the final leader board of the competition among 142 teams with a score above the organizers' baseline.

2 Overview

2.1 Embeddings for semantics and emotion

We use different kinds of embeddings that have been deemed effective in the literature in capturing not only the syntactic or semantic information of the words, but also their emotional content. We breifly describe them in this section.

GloVe, (Pennington et al., 2014) is a widely used pre-trained vector representation that captures fine-grained syntactic and semantic regularities. It has shown great success in word similarity tasks and Named Entity Recognition benchmarks.

ELMo, or Embeddings from Language Models, (Peters et al., 2018) are deep contextualized word representations. These representations enclose a polysemy encoding, i.e., they capture the variation in the meaning of a word depending on its context. The representations are learned functions of the input, pre-trained with deep bi-directional LSTM model. It has been shown to work well in practice on multiple language understanding tasks like question answering, entailment and sentiment analysis. In this work, our objective is to detect emotion accurately giving the context. Hence, employing such contextual embedding can be crucial.

DeepMoji (Felbo et al., 2017) is a pre-trained model containing rich representations of emotional content. It has been pre-trained on the task of predicting the emoji contained in the text using Bi-directional LSTM layers combined with an attention layer. A distant supervision approach was deployed to collect a massive (1.2 billion Tweets) dataset with diverse set of noisy emoji labels on which DeepMoji is pre-trained. This led to state-of-the art performance when fine-tuning DeepMoji on a range of target tasks related to sentiment, emotion and sarcasm.

2.2 Hierarchical RNN for context

One of the building component of our proposed model (see Figure 1) is the Hierarchical or Context recurrent encoder-decoder (HRED) (Sordoni et al., 2015). HRED architecture is used for encoding dialogue context in the task of multi-turn dialogue generation task (Serban et al., 2016). It has been proven to be effective in capturing the context information of dialogue exchanges. It contains two types of recurrent neural net (RNN) units: *encoder* RNN which maps each utterance to an utterance vector; *context* RNN which further processes the utterance vectors. HRED is expected to produce a better representation of the context in dialogues because the *context* RNN allows the model to represent the information exchanges between the two speakers.

2.3 BERT

BERT, the Bidirectional Encoder Representations for Transformers, (Devlin et al., 2018) is a pre-trained model producing context representations that can be very convenient and effective. BERT representations can be fine-tuned to many downstream NLP tasks by adding just one additional output layer for the target task, eliminating the need for engineering a specific architecture for a task. Using this setting, it has advanced the state-of-the-art performances in 11 NLP tasks. Using BERT in this work has slightly improved the final result, when we combine it with our HRLCE in an ensemble setting.

2.4 Importance Weighting

Importance Weighting (Sugiyama and Kawanabe, 2012) is used when label distributions between the training and test sets are generally different, which is the case of the competition datasets (Table 2). It corresponds to weighting the samples according to their importance when calculating the loss.

A supervised deep learning model can be regarded as a parameterized function $f(\boldsymbol{x}; \boldsymbol{\theta})$. The backpropagation learning algorithm through a differentiable loss is a method of *empirical risk minimization* (ERM). Denote $(\boldsymbol{x}_i^{tr}, y_i^{tr})$, $i \in [1 \dots n_{tr}]$

are pairs of training samples, testing samples are $(\boldsymbol{x}^{te}, y^{te})$, $i \in [1 \dots n_{te}]$.

The ratio $P(\boldsymbol{x})^{te}/P(\boldsymbol{x})^{tr}$ is referred as the *importance* of a sample $\boldsymbol{x}$. When the label distribution of training data and testing data are different: $P(\boldsymbol{x}^{te}) \neq P(\boldsymbol{x}^{tr})$, the training of the model $f_{\boldsymbol{\theta}}$ is then called under *covariate shift*. In such situation, the parameter $\hat{\boldsymbol{\theta}}$ should be estimated through *importance-weighted ERM*:

$$\arg\min_{\boldsymbol{\theta}} \Big[\frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} \frac{P(\boldsymbol{x}^{te})}{P(\boldsymbol{x}^{tr})} \text{loss}(y_i^{tr}, f(\boldsymbol{x}_i^{tr}; \boldsymbol{\theta}))\Big]. \tag{1}$$

3 Models

Denote the input $\boldsymbol{x} = [u_1, u_2, u_3]$, where u_i is the ith penultimate utterance in the dialogue. y is the emotion expressed in u_3 while giving u_1 and u_2 as context.

To justify the effectiveness of the modules in HRLCE, we propose two baseline models: SA-LSTM (SL) and SA-LSTM-DeepMoji (SLD). The SL model is part of the SLD model, while the later one composes the utterance encoder of our HRLCE. Therefore, we illustrate the models consecutively in Sections 3.1, 3.2, and 3.3.

3.1 SA-LSTM (SL)

Let $\boldsymbol{x}$ be the concatenation of u_1 ,u_2, and u_3. Hereby, $\boldsymbol{x} = [x_1, x_2, \cdots, x_n]$, where x_i is the ith word in the combined sequence. Denote the pre-trained GloVe model as G. As GloVe model can be directly used by looking up the word x_i, we can use $G(x_i)$ to represent its output. On the contrary, ELMo embedding is not just dependent on the word x_i, but on all the words of the input sequence. When taking as input the entire sequence $\boldsymbol{x}$, n vectors can be extracted from the pre-trained ElMo model. Denote the vectors as $\boldsymbol{E} = [E_1, E_2, \cdots, E_n]$. E_i contains both contextual and semantic information of word x_i. We use a two-layer bidirectional LSTM as the encoder of the sequence $\boldsymbol{x}$. For simplicity, we denote it as $LSTM^e$. In order to better represent the information of x_i, we use the concatenation of $G(x_i)$ and E_i as the feature embedding of x_i. Therefore, we have the following recurrent progress:

$$h_t^e = LSTM^e([G(x_t); E_t], h_{t-1}^e). \tag{2}$$

h_t^e is the hidden state of encoder LSTM at time step t, and $h_0^e = \mathbf{0}$. Let $\boldsymbol{h}_{\boldsymbol{x}}^e = [h_t^e, h_t^e, \cdots, h_t^e]$ be the n hidden states of encoder given the input $\boldsymbol{x}$. Self-attention mechanism has been proven to be effective in helping RNN dealing with dependency problems (Lin et al., 2017). We use the multi-head version of the self-attention (Vaswani et al., 2017) and set the number of channels for each head as 1. Denote the self-attention module as SA, it takes as input all the hidden states of the LSTM and summarizes them into a single vector. This process is represented as $h_{\boldsymbol{x}}^{sa} = SA(\boldsymbol{h}_{\boldsymbol{x}}^e)$. To predict the model, we append a fully connected (FC) layer to project $h_{\boldsymbol{x}}^{sa}$ on to the space of emotions. Denote the FC layer as $output$. Let $o_{\boldsymbol{x}}^{SL} = output(h_{\boldsymbol{x}}^{sa})$, then the estimated label of $\boldsymbol{x}$ is the $\arg\max_i(o_{\boldsymbol{x}}^{SL})$, where i is ith value in the vector $o_{\boldsymbol{x}}^{SL}$.

	F1	Happy	Angry	Sad	Harm. Mean
SL	Dev	0.6430	0.7530	0.7180	0.7016
	Test	0.6400	0.7190	0.7300	0.6939
SLD	Dev	0.6470	0.7610	0.7360	0.7112
	Test	0.6350	0.7180	0.7360	0.6934
HRLCE	Dev	0.7460	0.7590	0.8100	**0.7706**
	Test	0.7220	0.766	0.8180	**0.7666**
BERT	Dev	0.7138	0.7736	0.8106	0.7638
	Test	0.7151	0.7654	0.8157	0.7631

Table 1: Macro-F1 scores and its harmonic means of the four models

3.2 SA-LSTM-DeepMoji (SLD)

SLD is the combination of SA and DeepMoji. An SLD model without the output layer is in fact the utterance encoder of the proposed HRLCE, which is illustrated in the right side of Figure 1. Denote the DeepMoji model as D, when taking as input $\boldsymbol{x}$, the output is represented as $h_{\boldsymbol{x}}^d = D(\boldsymbol{x})$. We concatenate $h_{\boldsymbol{x}}^d$ and $h_{\boldsymbol{x}}^{sa}$ as the feature representation of sequence of $\boldsymbol{x}$. Same as SL, an FC layer is added in order to predict the label: $o_{\boldsymbol{x}}^{SLD} = output([h_{\boldsymbol{x}}^{sa}; h_{\boldsymbol{x}}^d])$.

3.3 HRLCE

Unlike SL and SLD, the input of HRLCE is not the concatenation of u_1, u_2, and u_3.

Following the annotation in Section 3.1 and 3.2, an utterance u_i is firstly encoded as $h_{u_i}^{sa}$ and $h_{u_i}^d$. We use another two layer bidirectional LSTM as the context RNN, denoted as $LSTM^c$. Its hidden states are iterated through:

$$h_t^c = LSTM^c([h_{u_t}^{sa}; h_{u_t}^d], h_{t-1}^c), \tag{3}$$

where $h_0^c = \mathbf{0}$. The three hidden states $\boldsymbol{h^c} = [h_1^c, h_2^c, h_3^c]$, are fed as the input to a self-attention layer. The resulting vector $SA(\boldsymbol{h^c})$ is also projected to the label space by an FC layer.

3.4 BERT

BERT (Section 2.3) can take as input either a single sentence or a pair of sentences. A "sentence" here corresponds to any arbitrary span of contiguous words. In this work, in order to fine-tune BERT, we concatenate utterances u_1 and u_2 to constitute the first sentence of the pair. u_3 is the second sentence of the pair. The reason behind such setting is that we assume that the target emotion y is directly related to u_3, while u_1 and u_2 are providing additional context information. This forces the model to consider u_3 differently.

4 Experiment

4.1 Data preprocessing

From the training data we notice that emojis are playing an important role in expressing emotions. We first use *ekphrasis* package (Baziotis et al., 2017) to clean up the utterances. *ekphrasis* corrects misspellings, handles textual emotions (e.g. ':)))'), and normalizes tokens (hashtags, numbers, user mentions etc.). In order to keep the semantic meanings of the emojis, we use the *emojis* package[1] to first convert them into their textual aliases and then replace the ":" and "_" with spaces.

4.2 Environment and hyper-parameters

We use PyTorch 1.0 for the deep learning framework, and our code in Python 3.6 can be accessed in GitHub[2]. For fair comparisons, we use the same parameter settings for the common modules that are shared by the SL, SLD, and HRLCE. The dimension of *encoder* LSTM is set to 1500 per direction; the dimension of *context* LSTM is set to 800 per direction. We use Adam optimizer with initial learning rate as 5e-4 and a decay ratio of 0.2 after each epoch. The parameters of DeepMoji are set to trainable. We use *BERT-Large* pre-trained model which contains 24 layers.

	happy	angry	sad	others	size
Train	14.07%	18.26%	18.11%	49.56%	30160
Dev	5.15%	5.44%	4.54%	84.86%	2755
Test	4.28%	5.57%	4.45%	85.70%	5509

Table 2: Label distribution of train, dev, and test set

[1]https://pypi.org/project/emoji/

[2]https://github.com/chenyangh/SemEval2019Task3

According to the description in (CodaLab, 2019), the label distribution for *dev* and *test* sets are roughly 4% for each of the emotions. However, from the *dev* set (Table 2) we know that the proportions of each of the emotion categories are better described as %5 each, thereby we use %5 as the empirical estimation of distribution $P(\boldsymbol{x}^{te})$. We did not use the exact proportion of *dev* set as the estimation to prevent the overfitting towards *dev* set. The sample distribution of the *train* set is used as $P(\boldsymbol{x}^{tr})$. We use *Cross Entropy* loss for all the aforementioned models, and the loss of the training samples are weighted according to Eq. 1.

4.3 Results and analysis

We run 9-fold cross validation on the training set. Each iteration, 1 fold is used to prevent the models from overfitting while the remaining folds are used for training. Therefore, every model is trained 9 times to ensure stability. The inferences over *dev* and *test* sets are performed on each iteration. We use the majority voting strategy to merge the results from the 9 iterations. The results are shown in Table 1. It shows that the proposed HRLCE model performs the best. The performance of SLD and SL are very close to each other, on the *dev* set, SLD performs better than SL but they have almost the same overall scores on the *test* set. The Macro-F1 scores of each emotion category are very different from each other: the classification accuracy for emotion *Sad* is the highest in most of the cases, while the emotion *Happy* is the least accurately classified by all the models. We also noticed that the performance on the *dev* set is generally slightly better than that on the *test* set.

5 Conclusions

Considering the competitive results generated by BERT, we combined BERT and our proposed model in an ensemble and obtained 0.7709 on the final test leaderboard. From a confusion matrix of our final submission, we notice that there are barely miss-classifications among the three categories (*Angry, Sad*, and *Happy*). For example, the emotion Sad is rarely miss-classified as "Happy" or "Angry". Most of the errors correspond to classifying the emotional utterances in the *Others* category. We think, as future improvement, the models need to first focus on the binary classification "Others" versus "Not-Others", then the "Not-Others" are classified in their respective emotion.

References

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada. Association for Computational Linguistics.

Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. 2019a. Understanding emotions in text using deep learning and big data. *Computers in Human Behavior*, 93:309–317.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019b. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

CodaLab. 2019. Semeval19 task 3: Emocontext. `https://competitions.codalab.org/competitions/19790#learn_the_details-data-set-format`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Paul Ekman. 1992. An argument for basic emotions. *Cognition & emotion*, 6(3-4):169–200.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Chenyang Huang, Osmar R. Zaiane, Amine Trabelsi, and Nouha Dziri. 2018. Automatic dialogue generation with expressed emotions. In *16th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, New Orleans, USA.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 task 1: Affect in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 1–17. Association for Computational Linguistics.

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Robert Plutchik. 2001. The nature of emotions: Human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice. *American Scientist*, 89(4):344–350.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562. ACM.

Masashi Sugiyama and Motoaki Kawanabe. 2012. *Machine learning in non-stationary environments: Introduction to covariate shift adaptation*. MIT press.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Ali Yaddolahi, Ameneh Gholipour Shahraki, and Osmar R. Zaiane. 2017. Current state of text sentiment analysis from opinion to emotion mining. *ACM Computing Surveys*, 50(2):25:1–25:33.

SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter

Valerio Basile[◇] **Cristina Bosco**[◇] **Elisabetta Fersini**[♡]
Debora Nozza[♡] **Viviana Patti**[◇] **Francisco Rangel**[♠♣]
Paolo Rosso[♣] **Manuela Sanguinetti**[◇]
[◇] Dipartimento di Informatica, Università degli Studi di Torino (Italy)
[♣] Università degli Studi di Milano Bicocca (Italy)
[♠] Autoritas Consulting (Spain)
[♡] PRHLT Research Center, Universitat Politècnica de València (Spain)
[◇]`{name.surname}@unito.it`, [♡]`{name.surname}@unimib.it`,
[♠]`francisco.rangel@autoritas.es`,
[♣]`prosso@dsic.upv.es`

Abstract

The paper describes the organization of the SemEval 2019 Task 5 about the detection of hate speech against immigrants and women in Spanish and English messages extracted from Twitter. The task is organized in two related classification subtasks: a main binary subtask for detecting the presence of hate speech, and a finer-grained one devoted to identifying further features in hateful contents such as the aggressive attitude and the target harassed, to distinguish if the incitement is against an individual rather than a group. HatEval has been one of the most popular tasks in SemEval-2019 with a total of 108 submitted runs for Subtask A and 70 runs for Subtask B, from a total of 74 different teams. Data provided for the task are described by showing how they have been collected and annotated. Moreover, the paper provides an analysis and discussion about the participant systems and the results they achieved in both subtasks.

1 Introduction

Hate Speech (HS) is commonly defined as any communication that disparages a person or a group on the basis of some characteristic such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics (Nockleby, 2000). Given the huge amount of user-generated contents on the Web, and in particular on social media, the problem of detecting, and therefore possibly contrasting the HS diffusion, is becoming fundamental, for instance for fighting against misogyny and xenophobia.
Some key aspects feature online HS, such as virality, or presumed anonymity, which distinguish it from offline communication and make it potentially also more dangerous and hurtful. Often hate speech fosters discrimination against particular categories and undermines equality, an everlasting issue for each civil society. Among the mainly targeted categories there are immigrants and women. For the first target, especially raised by refugee crisis and political changes occurred in the last few years, several governments and policy makers are currently trying to address it, making especially interesting the development of tools for the identification and monitoring such kind of hate (Bosco et al., 2017). For the second one instead, hate against the female gender is a long-time and well-known form of discrimination (Manne, 2017). Both these forms of hate content impact on the development of society and may be confronted by developing tools that automatically detect them.

A large number of academic events and shared tasks for different languages (i.e. English, Spanish, Italian, German, Mexican-Spanish, Hindi) took place in the very recent past which are centered on HS and related topics, thus reflecting the interest by the NLP community. Let us mention the first and second edition of the *Workshop on Abusive Language*[1] (Waseem et al., 2017), the *First Workshop on Trolling, Aggression and Cyberbullying* (Kumar et al., 2018), that also included a shared task on aggression identification, the tracks on *Automatic Misogyny Identification* (AMI) (Fersini et al., 2018b) and on *Authorship and Aggressiveness Analysis* (MEX-A3T) (Carmona et al., 2018) proposed at the 2018 edition of IberEval[2], the GermEval Shared Task on the *Identification of Offensive Language* (Wiegand et al.,

[1]`http://sites.google.com/view/alw2018/`
[2]`http://sites.google.com/view/ibereval-2018`

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 54–63
Minneapolis, Minnesota, USA, June 6–7, 2019. ©2019 Association for Computational Linguistics

2018), and finally the *Automatic Misogyny Identification task* (AMI) (Fersini et al., 2018a) and the *Hate Speech Detection task* (HaSpeeDe) (Bosco et al., 2018) at EVALITA 2018[3] for investigating respectively misogyny and HS in Italian.

HatEval consists in detecting hateful contents in social media texts, specifically in Twitter's posts, against two targets: immigrants and women. Moreover, the task implements a multilingual perspective where data for two widespread languages, English and Spanish, are provided for training and testing participant systems.
The motivations for organizing HatEval go beyond the advancement of the state of the art for HS detection for each of the involved languages and targets. The variety of targets of hate and languages provides a unique comparative setting, both with respect to the amount of data collected and annotated applying the same scheme, and with respect to the results achieved by participants training their systems on those data. Such comparative setting may help in shedding new light on the linguistic and communication behaviour against these targets, paving the way for the integration of HS detection tools in several application contexts. Moreover, the participation of a very large amount of research groups in this task (see Section 4) has improved the possibility of in-depth investigation of the involved phenomena.

The paper is organized as follows. In the next section, the datasets released to the participants for training and testing the systems are described. Section 3 presents the two subtasks and the measures we exploited in the evaluation. Section 4 reports on approaches and results of the participant systems. In Section 5, a preliminary analysis of common errors in top-ranked systems is proposed. Section 6 concludes the paper.

2 Data

The data have been collected using different gathering strategies. For what concerns the time frame, tweets have been mainly collected in the time span from July to September 2018, with the exception of data with target women. Indeed, the most part of the training set of tweets against women has been derived from an earlier collection carried out in the context of two previous challenges on misogyny identification (Fersini et al., 2018a,b). Different approaches were employed

	Training		**Test**	
Label	**Imm.**	**Women**	**Imm.**	**Women**
Hateful	39.76	44.44	42.00	42.00
Non-Hateful	60.24	55.56	58.00	58.00
Individual Target	5.89	64.94	3.33	80.63
Generic Target	94.11	35.06	96.67	19.37
Aggressive	55.08	30.06	59.84	34.44
Non-Aggressive	44.92	69.94	40.16	65.56

Table 1: Distribution percentages across sets and categories for English data. The percentages for the target and aggressiveness categories are computed on the total number of hateful tweets.

	Training		**Test**	
Label	**Imm.**	**Women**	**Imm.**	**Women**
Hateful	41.93	41.38	40.50	42.00
Non-Hateful	58.07	58.62	59.50	58.00
Individual Target	13.72	87.58	32.10	94.94
Generic Target	86.28	12.42	67.90	5.06
Aggressive	68.58	87.58	50.31	92.56
Non-Aggressive	31.42	12.42	46.69	7.44

Table 2: Distribution percentages across sets and categories for Spanish data. The percentages for the target and aggressiveness categories are computed on the total number of hateful tweets.

to collect tweets: (1) monitoring potential victims of hate accounts, (2) downloading the history of identified haters and (3) filtering Twitter streams with keywords, i.e. words, hashtags and stems. Regarding the keyword-driven approach, we employed both neutral keywords (in line with the collection strategy applied in Sanguinetti et al. (2018)), derogatory words against the targets, and highly polarized hashtags, in order to collect a corpus for reflecting also on the subtle but important differences between HS, offensiveness (Wiegand et al., 2018) and stance (Taulé et al., 2017). The keywords that occur more frequently in the collected tweets are: *migrant*, *refugee*, *#buildthatwall*, *bitch*, *hoe*, *women* for English, and *inmigra-*, *arabe*, *sudaca*, *puta*, *callate*, *perra* for Spanish[4].

The entire HatEval dataset is composed of 19,600 tweets, 13,000 for English and 6,600 for Spanish. They are distributed across the targets as follows: 9,091 about immigrants and 10,509 about women (see also Tables 1 for English and 2 for Spanish). Figures 1 and 2 show the distribution of the labels in the training and development set data according to the different targets of hate (woman and immigrants, respectively).

[3]http://evalita.org

[4]The complete set of keywords exploited is available here: https://github.com/msang/hateval/blob/master/keyword_set.md

2.1 Annotation

The data are released after the annotation process, which involved non-trained contributors on the crowdsourcing platform *Figure Eight* (F8)[5]. The annotation scheme applied to the HatEval data is a simplified merge of schemes already applied in the development of corpora for HS detection and misogyny by the organizers (Fersini et al., 2018a,b; Bosco et al., 2018), also in the context of funded projects with focus on the tasks topics[6] (Sanguinetti et al., 2018; Poletto et al., 2017). It includes the following categories:

- **HS** - a binary value indicating if HS is occurring against one of the given targets (women or immigrants): 1 if occurs, 0 if not.
- **Target Range** - if HS occurs (i.e. the value for the feature HS is 1), a binary value indicating if the target is a generic group of people (0) or a specific individual (1).
- **Aggressiveness** - if HS occurs (i.e. the value for the feature HS is 1), a binary value indicating if the tweeter is aggressive (1) or not (0).

We gave the annotators a series of guidelines in English and Spanish, including the definition for hate speech against the two targets considered, the aggressiveness's definition and a list of examples[7]. As requested by the platform, we provided a restricted set of "correct" answers to test the reliability of the annotators. We required to collect at least three independent judgments for each tweet. We adopted the default F8 settings for assigning the majority label (relative majority). The F8 reported average confidence (i.e., a measure combining inter-rater agreement and reliability of the contributor) on the English dataset for the fields HS, TR, AG is 0.83, 0.70 and 0.73 respectively, while for the Spanish dataset is 0.89, 0.47 and 0.47. The use of crowdsourcing has been successfully already experimented in several tasks and in HS detection too, both for English (Davidson et al., 2017) and other languages (Sanguinetti et al., 2018). However, stimulated by the discussion in (Basile et al., 2018), we decided to apply a similar methodology by adding two more expert annotations to all the crowd-annotated data, provided by native or near-native speakers of British English and Castilian Spanish, having a long experience in annotating data for the specific task's subject. We assigned the final label for this data based on majority voting from *crowd*, *expert1*, and *expert2*. This does not erase the contribution of the crowd, but hopefully maximises consistency with the guidelines in order to provide a solid evaluation benchmark for this task.

For data release and distribution each post has been identified by a newly generated index which substitutes the original Twitter's IDs.

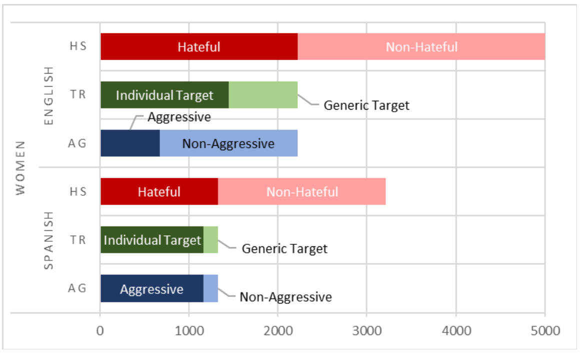


Figure 1: Distribution of the annotated categories in English and Spanish training and development set for the target women.

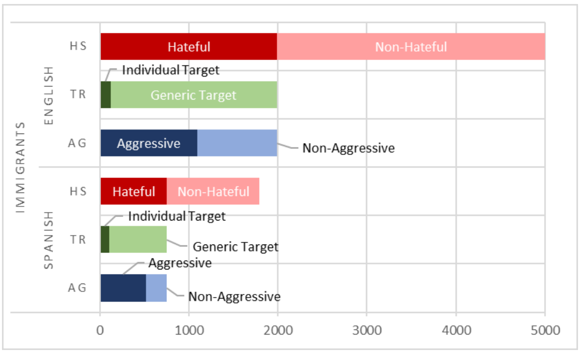


Figure 2: Distribution of the annotated categories in English and Spanish training and development set for the target immigrants.

2.2 Training, Development and Test Data

Data for training and development were released according to the distribution described in Figures 1 and 2 across languages (Spanish and English) and targets (women and immigrants). For what concerns Spanish, the training and development set includes 5,000 tweets, (3,209 for the target women and 1,991 for immigrants), while for English it in-

[5] http://www.figure-eight.com/

[6] http://hatespeech.di.unito.it/ihateprejudice.html.

[7] Annotation guidelines provided are accessible here: https://github.com/msang/hateval/blob/master/annotation_guidelines.md.

cludes 10,000 tweets (5,000 for each target). For a cross-language perspective see Figures 1 and 2. It can be also observed that the distribution across categories is pivoting around the main task category, HS, while the other ones more freely vary. Indeed, in order to provide a more balanced distribution of the HS and non-HS categories in the dataset released for Subtask A, we altered the natural distribution: both in the training and test set, hateful tweets are over-represented with respect to the distribution observed in the data we collected from Twitter[8]. Instead, the distribution of the other categories which are relevant for Subtask B is not constrained, and naturally follows from the selection of tweets for representing the classes relevant for the main Subtask A.

As far as the test set is concerned, 3,000 tweets have been annotated for English, half with target women and half immigrants, and 1,600 for Spanish distributed with the same proportion across the targets of hate: 1,260 hateful tweets and 1,740 non-hateful tweets for English, 660 hateful tweets and 940 non-hateful tweets for Spanish.

According to the schema described above, the format of an annotated tweet in the training and development set has the following pattern:

ID, Tweet-text, HS, TR, AG

where ID is a progressive number denoting the tweet within the dataset, Tweet-text is the given text of the tweet, while the other parts of the pattern, given in the training data and to be predicted in the test set, are: Hate Speech [HS] (1 or 0), Target Range [TR] (0 for group or 1 for individual), and Aggressiveness [AG] (0 or 1). Data included in the test instead only include ID and Tweet-text, the annotation of HS, TR and AG to be provided by participants according to the subtask.
An example of annotation is the following:

7, lol, chop her head off and rape the bitch https://t.co/ZB8CosmSD8, 1, 1, 1

which has been considered by the annotators as hateful, against an individual target, and aggressive. The latter category is not necessarily associated to HS, as shown in the following example, where a hateful content is expressed against a generic group of people in terms of disrespect and misogynistic stereotypes rather than using an aggressive language:

11, WOW can't believe all these women riding the subway today? Shouldn't these bitches be making sandwiches LOL #ihatefemales.., 1, 0, 0

3 Task Description

The task is articulated around two related subtasks. The first consists of a basic detection of HS, where participants are asked to mark the presence of hateful content. In the second subtask instead fine-grained features of hateful contents are investigated in order to understand how existing approaches may deal with the identification of especially dangerous forms of hate, i.e., those where the incitement is against an individual rather than against a group of people, and where an aggressive behaviour of the author can be identified as a prominent feature of the expression of hate. The participants will be asked in this latter subtask to identify if the target of hate is a single human or a group of persons, and if the message author intends to be aggressive, harmful, or even to incite, in various forms, to violent acts against the target (see e.g. (Sanguinetti et al., 2018)).

3.1 Subtask A - Hate Speech Detection against immigrants and women

Subtask A is a two-class (or binary) classification task where the system has to predict whether a tweet in English or in Spanish with a given target (women or immigrants) contains HS or not. The following sentences present examples of a hateful and non-hateful tweet where the targets are women.

[hateful]

[id: 32411] Cállate @[redacted] y la gran puta madre que te repario. Que le diste a la poltica...nada. Basura.
Shut up @[redacted] you motherfucker. What did you do for politics... nothing. Trash.[9]

[non-hateful]

[id: 33033] @[redacted] This is inhumane 😔 Karma is a bitch she ll get around these brainless heartless assholes!

[8]The whole original annotated dataset was very skewed towards the non-HS class (only about 10% of the annotated data contained hate speech).

[9]The target of the misogynistic hate here is Victoria Donda Prez, an Argentinian woman, human rights activist and member of the Argentine National Congress (mentioned in the at-mention of the original tweet).

3.2 Subtask B - Aggressive behaviour and Target Classification

Next, in Subtask B systems are asked to classify hateful tweets (e.g., tweets where HS against our targets has been identified) regarding both aggressive attitude and the target harassed. On one hand, the kind of target must be classified, and the task is binary:

- *Individual*: the text includes hateful messages purposely sent to a specific target.
- *Generic*: it refers to hateful messages posted to many potential receivers.

[**Individual**]:

> [id: 4723] @[redacted] Come on box I show you my cock darling

[**Generic**]:

> [id: 5823] Women are equal and deserve respect. Just kidding, they should suck my dick.

On the other hand, the aggressive behaviour has to be identified, then we propose a two-class classification task also for this feature. A tweet must be classified as aggressive or not:

[**Aggressive**]

> [id:1890] Sick barstewards! This is what happens when we put up the refugees welcome signs! They not only rape our wives or girlfriends, our daughters but our ruddy mothers too!! https://t.co/XAYLr6FjNk

[**Non-Aggressive**]

> [id: 945] @EmmanuelMacron Hello?? Stop groping my nation.Schneider: current migrant crisis represents a plan orchestrated and prepared for a long time by international powers to radically alter Christian and national identity of European peoples.http

3.3 Evaluation Measures and Baseline

The evaluation of the results considers different strategies and metrics for Subtasks A and B in order to allow more fine-grained scores.

Subtask A. Systems will be evaluated using standard evaluation metrics, including Accuracy, Precision, Recall and macro-averaged F_1-score.

In order to provide a measure that is independent on the class size, the submissions will be ranked by macro-averaged F_1-score, computed as described in (Özgür et al., 2005). The metrics will be computed as follows:

$$Accuracy = \frac{\textit{number of correctly predicted instances}}{\textit{total number of instances}} \quad (1)$$

$$Precision = \frac{\textit{number of correctly predicted instances}}{\textit{number of predicted labels}} \quad (2)$$

$$Recall = \frac{\textit{number of correctly predicted labels}}{\textit{number labels in the gold standard}} \quad (3)$$

$$F_1\text{-score} = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (4)$$

Subtask B. The evaluation of systems participating to Subtask B will be based on two criteria: (1) partial match and (2) exact match. Regarding the partial match, each dimension to be predicted (HS , TR and AG) will be evaluated independently from the others using standard evaluation metrics, including accuracy, precision, recall and macro-averaged F_1-score. We will report to the participants all the measures and a summary of the performance in terms of macro-averaged F_1-score, computed as follows:

$$F_1\text{-score} = \frac{F_1(HS) + F_1(AG) + F_1(TR)}{3} \quad (5)$$

Concerning the exact match, all the dimensions to be predicted will be jointly considered computing the Exact Match Ratio (Kazawa et al., 2005). Given the multi-label dataset consisting of n multi-label samples (x_i, Y_i), where x_i denotes the i-th instance and Y_i represents the corresponding set of labels to be predicted (HS $\in \{0,1\}$, TR $\in \{0,1\}$ and AG $\in \{0,1\}$), the Exact Match Ratio (EMR) will be computed as follows:

$$EMR = \frac{1}{n}\sum_{i=1}^{n} I(Y_i, Z_i) \quad (6)$$

where Z_i denotes the set of labels predicted for the i-th instance and I is the indicator function. The submissions will be ranked by EMR. This choice is motivated by the willingness to capture the difficulty of modeling the entire phenomenon, and therefore to identify the most dangerous behaviours against the targets.

Baselines. In order to provide a benchmark for the comparison of the submitted systems, we considered two different baselines. The first one (*MFC baseline*) is a trivial model that assigns the most frequent label, estimated on the training set, to all the instances in the test set. The second one (*SVC baseline*) is a linear Support Vector Machine (SVM) based on a TF-IDF representation, where the hyper-parameters are the default values set by the scikit-learn Python library (Pedregosa et al., 2011).

4 Participant Systems and Results

HatEval has been one of the most popular tasks in SemEval-2019 with a total of 108 submitted runs for Subtask A and 70 runs for Subtask B. We received submission from 74 different teams, of which 22 teams participated to all the subtasks for the two languages[10].

Besides traditional Machine Learning approaches, it has been observed that more than half of the participants investigated Deep Learning models. In particular, most of the systems adopted models known to be particularly suitable for dealing with texts, from Recurrent Neural Networks to recently proposed language models (Sabour et al., 2017; Cer et al., 2018). Consequently, external resources such as pre-trained Word Embeddings on tweets have been widely adopted as input features. Only a few works deepen the linguistic features analysis, probably due to the high expectations on the ability of Deep Learning models to extract high-level features. Most of the submitted systems adopted traditional preprocessing techniques, such as tokenization, lowercase, stopwords, URLs and punctuation removal. Some participants investigated Twitter-driven preprocessing procedures such as hashtag segmentation, slang conversion in correct English and emoji translation into words. It is worth mentioning that the construction of customized hate lexicons derived by the detection of language patterns in the training set has been preferred to the use of external hate lexicons expressing a more universal knowledge about the hate speech phenomenon, additionally demonstrating the need of developing more advanced approaches for detecting hate speech towards women and immigrants.

[10]The evaluation results are published here: https://docs.google.com/spreadsheets/d/1wSFKh1hvwwQIoY8_XBVkhjxacDmwXFpkshYzLx4bw-0/

4.1 Subtask A - Hate Speech Detection against immigrants and women

We received 69 submissions to the English Subtask A, of which 49% and 96% outperformed the SVC and MFC baseline respectively, in terms of macro-averaged F_1-score. Among the five best performing teams, only the team of *Panaetius*, which obtained the second position (0.571), has not provided a description of their system. The higher macro-averaged F_1-score (0.651) has been obtained by the Fermi team. They trained a SVM model with RBF kernel only on the provided data, exploiting sentence embeddings from Google's Universal Sentence Encoder (Cer et al., 2018) as features. Both the third, fourth and fifth ranked teams employ Neural Network models and, more specifically, Convolutional Neural Networks (CNNs) and Long Short Term Memory networks (LSTMs). In particular, the third position has been obtained by the *YNU_DYX* team, which system achieved 0.535 macro-averaged F_1-score by training a stacked Bidirectional Gated Recurrent Units (BiGRUs) (Cho et al., 2014) exploiting fastText word embeddings (Joulin et al., 2017). Then, the output of BiGRU is fed as input to the capsule network (Sabour et al., 2017). The textual preprocessing has been conducted with standard procedures, e.g. punctuation removal, tokenization, contraction normalization, use of tags for hyperlinks, numbers and mentions. The fourth place has been achieved by the team of *alonzorz* (0.535), which used a novel type of CNN called Multiple Choice CNN on the top of contextual embeddings. These embeddings have been created with a model similar to Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) trained using 50 million unique tweets from the Twitter Firehose dataset. The *SINAI-DL* team ranked fifth with a F_1-score of 0.519. They employ a LSTM model based on the pretrained GloVe Word Embeddings from Stanford-NLP group (Pennington et al., 2014). Since Deep Learning models require a large amount of data for training, they perform data augmentation through the use of paraphrasing tools. For preprocessing the texts in the specific Twitter domain, they convert all the mentions to a common tag and they tokenized hashtags according to the Camel Case procedure, i.e. the practice of writing phrases such that each word or abbreviation in the middle of the phrase begins with a capital letter, with no intervening spaces or punctuation.

For Subtask A in Spanish, we received 39 submissions of which 51% and 100% outperformed the SVC and MFC baseline respectively, in terms of macro-averaged F_1-score. The *Atalaya* and *MineriaUNAM* teams obtained the best macro-averaged F_1-score of 0.73, both taking advantage of Support Vector Machines. The *Atalaya* team studied several sophisticated systems, however the best performances have been obtained by a linear-kernel SVM trained on a text representation composed of bag-of-words, bag-of-characters and tweet embeddings, computed from fastText sentiment-oriented word vectors. The system proposed by the *MineriaUNAM* team is based on a linear-kernel SVM. The study has focused on a combinatorial framework used to search for the best feature configuration among a combination of linguistic patterns features, a lexicon of aggressive words and different types of n-grams (characters, words, POS tags, aggressive words, word jumps, function words and punctuation symbols). The *MITRE* team has achieved the performance of 0.729, presenting a novel method for adapting pre-trained BERT models to Twitter data using a corpus of tweets collected during the same time period of the HatEval training dataset. The *CIC-2* team achieved 0.727 with a word-based representation by combining Logistic Regression, Multinomial Naïve Bayes, Classifiers Chain and Majority Voting. They used TF and TF/IDF after removing HTML tags, punctuation marks and special characters, converting slang and short forms into correct English words and stemming. The participants did not use external resources and trained their systems only with the provided data. Finally, the *GSI-UPM* team obtained the macro-averaged F_1-score of 0.725 with a system where the linear-kernel SVM has been trained on an automated selection of linguistic and semantic features, sentiment indicators, word embeddings, topic modeling features, and word and character TF-IDF n-grams.

Table 3 shows basic statistics computed both for Subtasks A and B, with respect to the relative performance measures. The statistics comprise mean, standard deviation (StdDev), minimum, maximum, median and the first and third quartiles (Q1 and Q3). Concerning Subtask A, we notice that the maximum value in Spanish (0.7300) is higher than the English one (0.6510), while the difference is even higher (23 points) when considering the mean value, from 0.6821 to 0.4484. On the other hand, the variability is very similar between English (0.0569) and Spanish (0.0521).

	Subtask A		**Subtask B**	
	English	**Spanish**	**English**	**Spanish**
Min.	0.3500	0.4930	0.1590	0.4280
Q1	0.4050	0.6665	0.2790	0.5820
Mean	0.4484	0.6821	0.3223	0.6013
Median	0.4500	0.7010	0.3120	0.6160
StdDev	0.0569	0.0521	0.0890	0.0662
Q3	0.4880	0.7165	0.3570	0.6365
Max.	0.6510	0.7300	0.5700	0.7050
SVC Baseline	0.451	0.701	0.308	0.588
MFC Baseline	0.367	0.370	0.580	0.605

Table 3: Basic statistics of the results for the participating system and baselines in Subtask A and Subtask B expressed in terms of macro-averaged F_1-score and EMR respectively.

4.2 Subtask B - Aggressive behaviour and Target Classification

For Subtask B in English, we received 39 submissions, of which no system has been able to outperform the MFC baseline, which achieved 0.580 of EMR, while 61% outperformed the SVC baseline. Among the five best performing teams, only the team of *scmhl5*, which obtained the third position (0.483), has not provided us with a description of the system. The higher EMR result has been obtained by the *LT3* team with a value of 0.570. They considered a supervised classification-based approach with SVM models which combines a variety of standard lexical and syntactic features with specific features for capturing offensive language exploiting external lexicons. The second position has been obtained by the *CIC-1* team. The team achieved 0.568 in EMR with Logistic Regression and Classifier Chains. They trained their model only with the provided data, with a word-based representation and without external resources. The only preprocessing action was stemming and stop words removal. The fourth position was obtained by the team named The Titans. They achieved 0.471 of EMR with LSTM and TF/IDF-based Multilayer Perceptron. To represent the documents, they used the tweet words after removing links, mentions and spaces. They also tokenized hashtags into word tokens. The MITRE team exploits the same approach used for participating in Subtask A, obtaining 0.399 EMR. It is worth mentioning that, despite the fact that the baseline could not be overcome in terms of EMR, the five first performing systems obtained higher F-values. For example, while the baseline obtained 0.421, the *scmhl5* (0.632) and the *MITRE* team (0.614) systems obtained about 20 points over it.

For Subtask B in Spanish, we received 23 submissions of which 52% and 70% outperformed the SVC and MFC baseline respectively, in terms of EMR. The first position has been achieved by the CIC-2 team with 0.705 in terms of EMR, proposing the same approach for Subtask A in Spanish. The *CIC-1* and *MITRE* teams, described previously, achieved the second and third positions with 0.675 and 0.675 in EMR respectively. The fourth position was obtained by the *Atalaya* team that achieved 0.657 EMR by extending the previously presented approach for Subtask A to a 5-way classification problem for all the possible label combinations. Finally, the team of *Oscar-Garibo* achieved the fifth position (0.6444) with Support Vector Machines and statistical embeddings to represent the texts. The proposed method, a variation of LDSE (Rangel et al., 2016), consists of finding thresholds on the frequencies of use of the different terms in the corpora depending on the class they belong to. In this subtask, the correlation between EMR and macro-averaged F_1-score is more homogeneous than in English. However, it is worth mentioning the case of the *CIC-1* team since its macro-averaged F_1-score decreases with respect to the EMR and is 10 points lower than the rest of the best five performing teams.

The comparative results between all the performing teams in the two languages show interesting insights (see Table 3). Firstly, the best result is much higher in the case of Spanish (0.7050) than in English (0.5700) in more than 13 points. In the case of the fifth best results, the difference is much higher (0.2454), from 0.3990 in English to 0.6440 in Spanish. The average value changes from 0.3223 in English to 0.6013 in Spanish, with a difference of 28 points. The variability is also higher in English (0.0890) with respect to the value in Spanish (0.0662).

We can also derive further conclusions by comparing the statistics of the two Subtasks. Looking at the median, it is possible to notice that in both languages, the performances obtained on Subtask B are lower than the performances of Subtask A, with a difference between Subtask A and B of 14 and 8 points for English and Spanish respectively. This suggests that participant systems found much harder to predict the aggressiveness and targets than just the presence of hate speech. The quartile Q1 has highlighted that for the English language 75% of the systems obtained a score higher than 0.41 and 0.28 for Subtasks A and B, in particular 50 out of 69 for Subtask A and 31 out of 41 for Subtask B. While Q3 shows that 25% of the systems achieved a score value higher than 0.49 and 0.36 for Subtasks A and B, in particular 18 out of 69 for Subtask A and 11 out of 41 for Subtask B. For the Spanish language, the value of Q1 indicates that 75% of the systems have a score higher than 0.67 and 0.58 for Subtasks A and B, in particular 30 out of 39 for Subtask A and 17 out of 23 for Subtask B. Observing the quartile Q3, it is possible to observe that 25% of the systems achieved a value higher than 0.72 and 0.64 for Subtasks A and B, in particular 10 out of 39 for Subtask A and 6 out of 23 for Subtask B. Moreover, it is worth mentioning that the smaller the standard deviation the closer are the data to the mean value, highlighting that the Subtask B has shown high variability in terms of results than Subtask A. This statistics remarks again the difficulties of addressing Subtask B compared to Subtask A.

5 Error Analysis

In order to gain deeper insight into the results of the HatEval evaluation, we conducted a first error analysis experiment. For both languages, we selected the three top-ranked systems and checked the instances in the test set that were wrongly labeled by all three of them.

In the English Subtask A, the three top systems (*Fermi*, *Panaetius*, and *YNU_DYX*) predicted the same wrong labels 569 times out of 2,971 (19.1%). In the Spanish Subtask A, the three top systems (*Atalaya*, *mineriaUNAM*, and *MITRE*) predicted the same wrong labels 234 times out of 1,600 (14.6%). The results showing the percentages by wrongly assigned labels are summarized in Table 4.

Subtask	**Errors**	**Predicted 1**	**Predicted 0**
EN A	569	507 (89.1%)	62 (10.9%)
ES A	234	178 (76.1%)	56 (23.9%)

Table 4: Number of instances mislabeled by all the three top-ranked systems, broken down by wrongly assigned label.

The common errors are highly skewed towards the false positives. However, the unbalance is stronger for English (89.1% false positives) than for Spanish (76% false positives).

Two English examples, respectively a false positive and a false negative, are:

> [id: 30249] My mom FaceTimed me to show off new shoes she got and was like "no cabe duda que soy una Bitch" i love her 😂

> [id: 30542] @[redacted] @[redacted] There are NO INNOCENT people in detention centres #SendThemBack

The false positive contains a swear word ("Bitch") used in a humorous, not offensive context, which is a potential source of confusion for a classifier. The false negative is a hateful message towards migrants, but phrased in a slightly convoluted way, in particular due to the use of negation ("no innocent people").

Similarly, a false positive and a false negative in Spanish:

> [id: 33119] Soy un sudaca haciendo sudokus 🙃 https://t.co/vA7nQsfm85
> *I am a sudaca doing sudokus*

> [id: 34455] Estoy escuchando una puta canción y la pelotuda de Demi Lovato se pone a hablar en el medio. CANTÁ Y CALLATE LA BOCA.
> *I am listening to a fucking song and that asshole Demi Lovato starts talking in the middle of it. SING AND SHUT YOUR MOUTH.*

Like in the English example, in this false positive a negative word ("sudaca") is used humorously, for the purpose of a wordplay. In the false negative, there a misogynistic message is expressed, although covertly, implying that the target should "shut up and sing".

6 Conclusion

The very high number of participating teams at HatEval 2019 confirms the growing interest of the community around abusive language in social media and hate speech detection in particular. The presence of this task at SemEval 2019 was indeed very timely and the multilingual perspective we applied by developing data in two different widespread languages, English and Spanish, contributed to include and raise interest in a wider community of scholars. 38 teams sent their system reports to describe the approaches and the details of their participation to the task, contributing in shedding light on this difficult task. Some of the HatEval participants also participated to the OffensEval[11], another task related to abusive language identification, but with an accent on the different notion of *offensiveness*, an orthogonal notion that can characterize also expressions that cannot be featured as hate speech[12]. Overall, results confirm that hate speech detection against women and immigrants in micro-blogging texts is challenging, with a large room for improvement. We hope that the dataset made available as part of the shared task will foster further research on this topic, including its multilingual perspective.

Acknowledgments

Valerio Basile, Cristina Bosco, Viviana Patti and Manuela Sanguinetti are partially supported by Progetto di Ateneo/CSP 2016 (*Immigrants, Hate and Prejudice in Social Media*, S1618_L2_BOSC_01).

References

Valerio Basile, Nicole Novielli, Danilo Croce, Francesco Barbieri, Malvina Nissim, and Viviana Patti. 2018. Sentiment polarity classification at evalita: Lessons learned and open challenges. *IEEE Transactions on Affective Computing*.

Cristina Bosco, Felice Dell'Orletta, Fabio Poletto, Manuela Sanguinetti, and Maurizio Tesconi. 2018. Overview of the EVALITA 2018 Hate Speech Detection Task. In *Proceedings of the Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*. CEUR-WS.org.

Cristina Bosco, Patti Viviana, Marcello Bogetti, Michelangelo Conoscenti, Giancarlo Ruffo, Rossano Schifanella, and Marco Stranisci. 2017. Tools and Resources for Detecting Hate and Prejudice Against Immigrants in Social Media. In *Proceedings of First Symposium on Social Interactions in Complex Intelligent Systems (SICIS), AISB Convention 2017, AI and Society*.

Miguel Ángel Álvarez Carmona, Estefanía Guzmán-Falcón, Manuel Montes-y-Gómez, Hugo Jair Escalante, Luis Villaseñor Pineda, Verónica Reyes-Meza, and Antonio Rico Sulayes. 2018. Overview

[11] https://competitions.codalab.org/competitions/20011

[12] See (Sanguinetti et al., 2018) for a deeper reflection on hate speech and offensiveness.

of MEX-A3T at IberEval 2018: Authorship and Aggressiveness Analysis in Mexican Spanish Tweets. In *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018)*. CEUR-WS.org.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. *CoRR*, abs/1803.11175.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. *CoRR*, abs/1703.04009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Elisabetta Fersini, Debora Nozza, and Paolo Rosso. 2018a. Overview of the EVALITA 2018 Task on Automatic Misogyny Identification (AMI). In *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*. CEUR-WS.org.

Elisabetta Fersini, Paolo Rosso, and Maria Anzovino. 2018b. Overview of the Task on Automatic Misogyny Identification at IberEval 2018. In *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018)*. CEUR-WS.org.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 427–431.

Hideto Kazawa, Tomonori Izumitani, Hirotoshi Taira, and Eisaku Maeda. 2005. Maximal margin labeling for multi-topic text categorization. In *Advances in Neural Information Processing Systems*, pages 649–656.

Ritesh Kumar, Atul Kr. Ojha, Marcos Zampieri, and Shervin Malmasi, editors. 2018. *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*. ACL.

Kate Manne. 2017. *Down Girl. The Logic of Misogyny*. Oxford University Press.

John T. Nockleby. 2000. Hate speech. *Encyclopedia of the American Constitution (2nd ed., edited by Leonard W. Levy, Kenneth L. Karst et al., New York: Macmillan, 2000)*, pages 1277–1279.

Arzucan Özgür, Levent Özgür, and Tunga Güngör. 2005. Text categorization with class-based and corpus-based keyword selection. In *International Symposium on Computer and Information Sciences*, pages 606–615. Springer.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Fabio Poletto, Marco Stranisci, Manuela Sanguinetti, Viviana Patti, and Cristina Bosco. 2017. Hate Speech Annotation: Analysis of an Italian Twitter Corpus. In *Proceedings of the Fourth Italian Conference on Computational Linguistics (CLiC-it 2017)*. CEUR-WS.org.

Francisco Rangel, Paolo Rosso, and Marc Franco-Salvador. 2016. A low dimensionality representation for language variety identification. In *17th International Conference on Intelligent Text Processing and Computational Linguistics, CICLing*. Springer-Verlag, LNCS.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *Advances in neural information processing systems*, pages 3856–3866.

Manuela Sanguinetti, Fabio Poletto, Cristina Bosco, Viviana Patti, and Marco Stranisci. 2018. An Italian Twitter Corpus of Hate Speech against Immigrants. In *Proceedings of the 11th Language Resources and Evaluation Conference 2018*.

Mariona Taulé, Maria Antònia Martí, Francisco M. Rangel Pardo, Paolo Rosso, Cristina Bosco, and Viviana Patti. 2017. Overview of the task on stance and gender detection in tweets on catalan independence. In *Proceedings of the Second Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2017)*. CEUR-WS.org.

Zeerak Waseem, Wendy Hui Kyong Chung, Dirk Hovy, and Joel Tetreault, editors. 2017. *Proceedings of the First Workshop on Abusive Language Online*. ACL.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018)*.

Atalaya at SemEval 2019 Task 5: Robust Embeddings for Tweet Classification

Juan Manuel Pérez
Universidad de Buenos Aires
CONICET
jmperez@dc.uba.ar

Franco M. Luque
Universidad Nacional de Córdoba
CONICET
francolq@famaf.unc.edu.ar

Abstract

In this article, we describe our participation in HatEval, a shared task aimed at the detection of hate speech against immigrants and women. We focused on Spanish subtasks, building from our previous experiences on sentiment analysis in this language. We trained linear classifiers and Recurrent Neural Networks, using classic features, such as bag-of-words, bag-of-characters, and word embeddings, and also with recent techniques such as contextualized word representations. In particular, we trained robust task-oriented subword-aware embeddings and computed tweet representations using a weighted-averaging strategy. In the final evaluation, our systems showed competitive results for both Spanish subtasks ES-A and ES-B, achieving the first and fourth places respectively.

1 Introduction

Hate speech against women, immigrants, and many other groups is a pervasive phenomenon on the Internet. On the early days of the World Wide Web, many academics adventured that prejudices and hatred would be removed in this space by the dissolution of identities (Lévy, 2001; Rheingold, 1993). Twenty years after this hypothesis, we can say that it has not been the case. The prevalence of racism in the "World White Web" has been studied in a number of works (Adams and Roscigno, 2005; Kettrey and Laster, 2014) and so has been the misogyny in the virtual world (Filipovic, 2007; Mantilla, 2013).

Racist and sexist discourse are a constant in social media, but peaks are documented after "trigger" events, such as murders with religious or political reasons (Burnap and Williams, 2015). Most social media companies are concerned about this issue and take actions against it; nonetheless, most of the efforts still need human intervention, making this task very expensive. Therefore, reducing human intervention is vital in order to have effective tools to avoid the escalation of hate speech.

HatEval (Basile et al., 2019) is a SemEval-2019 shared task aimed at the detection of hate speech towards immigrants and women in tweets. It comprises two subtasks, with datasets in English (EN) and Spanish (ES) for both of them, giving a total of four subtasks. Subtask A is the binary classification of tweets into hateful or not hateful (HS). Subtask B is a triple binary classification task where, in addition to HS, tweets are classified into aggressive or not aggressive (AG), and targets of hate speech are classified into single humans or groups of persons (TR).

In this article, we present our participation in HatEval as team Atalaya. We focused our efforts on subtask A for Spanish (ES-A) but also worked at subtask B in Spanish (ES-B) and subtask A in English (EN-A). Our systems are based on our participation in the polarity classification task of Spanish tweets TASS 2018 (*Sentiment Analysis at SEPLN*) (Martínez-Cámara et al., 2018; Luque and Pérez, 2018).

To represent tweets, we experimented with a mixed approach of bag-of-words, bag-of-characters and tweet embeddings, which were calculated from word vectors using different averaging schemes. We used *fastText* (Bojanowski et al., 2016) to get subword-aware representations specifically trained for sentiment analysis tasks.

These word representations are robust to noise since they can be computed for unseen words by using subword embeddings. Moreover, we trained them using a database of 90M tweets from various Spanish-speaking countries, giving wide domain-specific vocabulary coverage. We achieved additional robustness by doing preprocessing using several text-normalization and noise-reduction techniques.

Also, we experimented with *ELMo* (Peters et al., 2018), a deep contextualized word representation that has drawn a lot of attention in the last months. Unlike *fastText*, *ELMo* returns context-dependent embeddings from a multi-layer bidirectional-LSTM language model. These representations improved the state-of-the-art of several NLP tasks.

For the neural approach, we used bidirectional LSTMs to combine the word embeddings. We also did experiments that mix sequential models with complementary representations such as bag-of-words.

The rest of the paper is as follows. Next Section presents the primary tools we used to build our systems. Section 3 presents the configuration and development of both linear and neural models. Section 4 briefly shows our results in the competition, and Section 5 concludes the work with some observations about our experience.

1.1 Previous Work

The detection of hate speech is a sentence classification task quite related to sentiment analysis and has been studied for several social media networks (Thelwall, 2008; Pak and Paroubek, 2010; Saleem et al., 2017). Regarding the detection of hateful content, Greevy and Smeaton (2004) used bag-of-words and SVMs to detect racist content in web pages. Following a similar approach, Warner and Hirschberg (2012) used unigrams and Brown clusters with SVMs to detect anti-semitic messages on Twitter.

Waseem and Hovy (2016) annotated a corpus and used character n-grams to detect hateful comments, and Badjatiya et al. (2017) used the same dataset to train deep learning models and fine-tuned embeddings along with Gradient Boosted Trees. Zhang et al. (2018) trained a deep neural network combining CNNs with Gated-recurrent units (Cho et al., 2014), outperforming previous systems in several datasets.

Anzovino et al. (2018) collected a corpus of misogynous tweets and proposed a taxonomy to distinguish them into different categories. The authors proposed a number of different techniques to classify them, showing that simple approaches (as using linear models along with token n-grams) achieve competitive performance on small-sized datasets.

Regarding shared tasks, Fersini et al. (2018a) presented a challenge on misogyny detection on Twitter –both in Spanish and English– whereas Fersini et al. (2018b) posed a similar challenge but in Italian and English. Bosco et al. (2018) proposed an automatic detection contest over Twitter posts and Facebook comments, comprising general hate speech.

2 Techniques and Resources

2.1 Preprocessing

Preprocessing is crucial in NLP applications, especially when working with noisy user-generated data. Here, we followed Luque and Pérez (2018), defining two levels of preprocessing: basic and sentiment-oriented preprocessing. We used one or the other, depending on the configuration.

Basic tweet preprocessing includes tokenization, replacement of handles, URLs, and e-mails, and shortening of repeated letters.

Sentiment-oriented preprocessing includes lowercasing, removal of punctuation, stopword, and numbers, lemmatization –using TreeTagger (Schmid, 1995)– and negation handling. For negation handling, we followed a simple approach: We find negation words and add the prefix `'NOT_'` to the following tokens. Up to three tokens are negated, or less if a non-word token is found.

2.2 Bags of Words and Characters

The simplest approach considered to build tweet representations was bag-of-words encoding. A bag-of-words (BoW) builds feature vectors for each token seen in training data. For a particular tweet, its BoW vector contains the number of occurrences of each token on it, resulting in high-dimensional and sparse vectors. Variations of BoW include counting not only single tokens but also n-grams of tokens, binarizing counts, and limiting the number of features.

Character usage in tweets may also hold useful information for sentiment analysis. Character n-grams –such as the presence and repetition of uppercase letters, emoticons, and exclamation marks– may indicate a strong presence of sentiment of some kind, where others may indicate a more formal writing style, and therefore an absence of sentiment.

To capture this information, we considered a bag-of-characters (BoC) representation that encodes counts of character n-grams for some values of n. These vectors are computed from original texts of tweets, with no preprocessing at all. BoCs have the same variants and parameters as BoWs.

2.3 Word Embeddings

We used *fastText*, a subword-aware embeddings library (Bojanowski et al., 2016) to get context-independent word representations. Instead of using publicly available pre-trained vectors, we trained our own embeddings on a dataset of $\sim$ 90 million tweets from various Spanish-speaking countries. We prepared two versions of the data: one using only basic preprocessing, and the other using sentiment-oriented preprocessing (with the exception of excepting lemmatization). For these two datasets, skip-gram embeddings were trained using different parameter configurations, including a number of dimensions, size of word and subword n-grams, and size of context window.

2.4 Tweet Embeddings

Linear combinations were used to compute a representation for a single tweet. We followed two simple approaches: plain average and weighted average. In the second case, we used a scheme that resembles Smooth Inverse Frequency (SIF) (Arora et al., 2017), inspired by TF-IDF reweighting. Each word w is weighted with $\frac{a}{a+p(w)}$, where $p(w)$ is the word unigram probability, and a is a smoothing hyper-parameter. Big values of a mean more smoothing towards plain averaging.

2.5 Context-Dependent Embeddings

After the great leap forward that represented context-independent word embeddings, a new wave came in the last years. Instead of having vectors trained for each word, context-dependent representations are generated for each token given a sentence. For instance, McCann et al. (2017) used a deep LSTM encoder for Machine Translation to generate context-aware vectors.

ELMo (Peters et al., 2018) is one of these context-dependent approaches and is based on a deep bidirectional language model (biLM). The architecture of the language model consists of L layers of bidirectional LSTMs, plus a context-independent token representation. Hence, for each token in a sequence, we get $2L + 1$ vector representations. To obtain a final vector for each token, the authors suggest collapsing the layers into vectors by means of a linear combination.

In this work, we used the implementation and pre-trained models from Che et al. (2018). The Spanish model was trained with $L = 2$ layers and 1024 dimensions, and the linear combination was done using a simple average.

3 Models

In this section, we describe the models we used in the competition.

3.1 Linear Classifiers

The first set of models we trained were simple classifying models implemented with scikit-learn (Pedregosa et al., 2011).

We started from the optimal configuration from Luque and Pérez (2018), that combines bag-of-words (BoW), bag-of-characters (BoC) and tweet embeddings as follows:

- BoW: All unigrams and bigrams of words, with binarized counts and TF-IDF reweighting. For the Spanish training dataset, this encoding gives 53504 sparse features.
- BoC: All n-grams of characters for $n \leq 5$, with binarized counts and TF-IDF reweighting. For the Spanish training dataset, it gives 226156 sparse features.
- Tweet embeddings: Computed from *fastText* sentiment-oriented word vectors of 50 dimensions. Weighted averaging was done as described in Section 2.4, with a smoothing value of $a = 0.1$.

Here, the only parameters specifically optimized using the HatEval development set were the n-gram ranges considered for BoW and BoC.

Using this vectorial representation we trained logistic regressions and linear-kernel SVMs with different hyperparameter values. The best results are shown in the first block of Tab. 1, as LR_0 and SVM_0.

Next, to confirm the relevance of each of the three components, we performed ablation tests for each of them. Results are displayed as SVM_{BoW}, SVM_{BoC} and SVM_{emb} in Tab. 1. Drops in the performance show the relevance of all components, especially for BoW and BoC.

Next, we tried adding tweet representations computed from ELMo vectors. Full tweet vectors were obtained by doing simple un-weighted averaging. PCA was optionally used to reduce the dimension of final vectors. The best results were

Model	Acc	F1 (avg)
LR_0	0.84	0.84
SVM_0	**0.85**	**0.85**
SVM_{BoW}	0.81	0.81
SVM_{BoC}	0.81	0.81
SVM_{emb}	0.84	0.84
SVM_{ELMo}	0.84	0.84

Table 1: Experiments with logistic regressions (LRs) and SVMs on the Spanish development set. Models are described in Section 3.1. The best result is in bold.

obtained using PCA to reduce from the original 1024 to 100 dimensions.

Results are shown as SVM_{ELMo} in Tab. 1. It can be seen that, under this configuration, we are not able to improve our results using ELMo.

To participate in the Spanish subtask B (ES-B) we used a very naive approach. We didn't develop or tune a specific system for this subtask but instead used the same system and configuration that was found optimal for subtask A. To do this, we first mapped the triple classification problem to a 5-way classification problem for all the possible label combinations:

HS	AG	TR
0	0	0
1	0	0
1	0	1
1	1	0
1	1	1

Then, we simply trained the classifier using the Spanish subtask B training dataset.

3.2 Neural Models

The second set of models we trained are neural models. We trained Recurrent Neural Networks (RNNs) using pre-trained context-dependent representations for Spanish.

The first model considered was a bidirectional LSTM with a dense layer on top, consuming ELMo vectors; we call this model *LSTM-ELMo*. Also, we tried another model by adding a second input consisting of a bag-of-words, as illustrated in Figure 1. We call this model *LSTM-ELMo+BoW*. Using *fastText* embeddings (of dimension 300 and context window 5) instead of BoW was considered as suggested by Peters et al. (2018) but discarded as it had no positive impact in performance (in the development dataset).

The *biLSTM* layer consists of 256 units. The bag-of-words has the 3500 most-frequent n-grams (having document-frequency less than 0.65), followed by a 512-unit dense layer. The two last dense layers have 64 neurons.

We used Keras (Chollet et al., 2015) to implement and train our models. *Adam* (Kingma and Ba, 2014) was the chosen optimizer, with $lr = 35 * 10^{-5}$ and $decay = 0.01$. To regularize our models, we applied dropout with keep-prob of 0.2 on the first layer, and 0.45 on the second, and we also early-stopped the training monitoring the performance on the development dataset. The hyperparameters were chosen from a small random search, as training ELMo is computationally expensive.

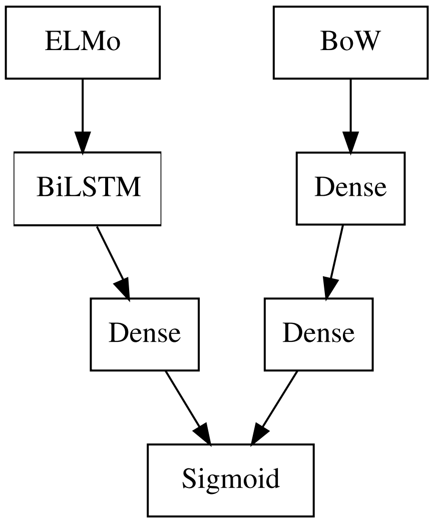


Figure 1: The *LSTM-ELMo+BoW* architecture. ELMo and BoW boxes represent inputs.

4 Results

Table 2 displays the evaluation results for the three classifiers trained for subtask A: SVM_0, and both neural models *LSTM-ELMo* and *LSTM-ELMo+BoW*. For Spanish, the best performing system was SVM_0. Despite its simplicity, it ranked first in terms of average F1 in the official results.

Among the neural models, *LSTM-ELMo+BoW* performed best, and ranked in position 17 for Spanish in terms of average F1.[1] We can observe that *LSTM-ELMo+BoW* performs better on the development set, although its performance decreases sharply in the test set. In spite of the applied regularization, we might have incurred in overfitting during model selection (Cawley and Talbot, 2010) as the chosen model has higher variance

[1]Results shown in Tab. 2 differ from the ones in the leaderboard as we couldn't exactly reproduce the experiments.

	Spanish				English			
	Dev		Test		Dev		Test	
Classifier	Acc	F1 (avg)	Acc	F1 (avg)	Acc	F1 (avg)	Acc	F1 (avg)
SVM_0	**0.850**	**0.850**	0.731	**0.730**	—	—	—	—
LSTM-ELMo	0.820	0.816	**0.732**	0.721	0.705	0.695	**0.508**	**0.471**
LSTM-ELMo+BoW	0.824	0.821	0.719	0.712	**0.743**	**0.738**	0.502	0.461

Table 2: Our evaluation results for subtask A on the development and test sets for Spanish and English. F1 (avg) is the average on positive and negative classes.

than *LSTM-ELMo*. This last model achieved similar results to SVM_0. This difference between the models was not seen in English.

For the Spanish subtask B (ES-B), the same SVM_0 system was used, achieving an average F1 of 0.758 and an EMR score of 0.657 over the test set (fourth place in terms of EMR).

5 Conclusion and future work

As in our previous experience with sentiment analysis, we found that linear models can be a match for neural models. Moreover, this time our SVM ranked in the first place in one of the subtasks.

We believe that –for this kind of challenges with small-sized datasets– preprocessing techniques, data normalization and robustness play a stronger role than model design and hyperparameter tuning. On the other hand, deep neural models are highly expressive and prone to overfitting, requiring being extremely careful with regularization.

Acknowledgments

We are grateful to Pablo Brusco for providing us with helpful comments. This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-18-1-0026, and also by a research grant from SeCyT, Universidad Nacional de Córdoba.

References

Josh Adams and Vincent J Roscigno. 2005. White supremacists, oppositional culture and the world wide web. *Social Forces*, 84(2):759–778.

Maria Anzovino, Elisabetta Fersini, and Paolo Rosso. 2018. Automatic identification and classification of misogynistic language on twitter. In *International Conference on Applications of Natural Language to Information Systems*, pages 57–64. Springer.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations*.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Cristina Bosco, Dell'Orletta Felice, Fabio Poletto, Manuela Sanguinetti, and Tesconi Maurizio. 2018. Overview of the evalita 2018 hate speech detection task. In *EVALITA 2018-Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian*, volume 2263, pages 1–9. CEUR.

Pete Burnap and Matthew L Williams. 2015. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet*, 7(2):223–242.

Gavin C Cawley and Nicola LC Talbot. 2010. On overfitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11(Jul):2079–2107.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64, Brussels, Belgium. Association for Computational Linguistics.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078.*

François Chollet et al. 2015. Keras. `https://keras.io`.

Elisabetta Fersini, Maria Anzovino, and Paolo Rosso. 2018a. Overview of the task on automatic misogyny identification at ibereval. In *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018), co-located with 34th Conference of the Spanish Society for Natural Language Processing (SEPLN 2018). CEUR Workshop Proceedings. CEUR-WS. org, Seville, Spain.*

Elisabetta Fersini, Debora Nozza, and Paolo Rosso. 2018b. Overview of the evalita 2018 task on automatic misogyny identification (ami). *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18), Turin, Italy. CEUR. org.*

Jill Filipovic. 2007. Blogging while female: How internet misogyny parallels real-world harassment. *Yale JL & Feminism*, 19:295.

Edel Greevy and Alan F Smeaton. 2004. Classifying racist texts using a support vector machine. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 468–469. ACM.

Heather Hensman Kettrey and Whitney Nicole Laster. 2014. Staking territory in the "world white web" an exploration of the roles of overt and color-blind racism in maintaining racial boundaries on a popular web site. *Social Currents*, 1(3):257–274.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Pierre Lévy. 2001. *Cyberculture*, volume 4. U of Minnesota Press.

Franco M. Luque and Juan Manuel Pérez. 2018. Atalaya at TASS 2018: Sentiment analysis with tweet embeddings and data augmentation. In *Proceedings of TASS 2018: Workshop on Semantic Analysis at SEPLN, TASS@SEPLN 2018, co-located with 34nd SEPLN Conference (SEPLN 2018), Sevilla, Spain, September 18th, 2018.*, pages 29–35.

Karla Mantilla. 2013. Gendertrolling: Misogyny adapts to new media. *Feminist Studies*, 39(2):563–570.

Eugenio Martínez-Cámara, Yudivián Almeida Cruz, Manuel C. Díaz-Galiano, Suilan Estévez Velarde, Miguel Á. García-Cumbreras, Manuel García-Vega, Yoan Gutiérrez Vázquez, Arturo Montejo Ráez, André Montoyo Guijarro, Rafael Muñoz Guillena, Alejandro Piad Morffis, and Julio Villena-Román. 2018. Overview of TASS 2018: Opinions, health and emotions. In *Proceedings of TASS 2018: Workshop on Semantic Analysis at SEPLN (TASS 2018)*, volume 2172 of *CEUR Workshop Proceedings*, Sevilla, Spain. CEUR-WS.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 1320–1326.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Howard Rheingold. 1993. *The virtual community: Finding commection in a computerized world.* Addison-Wesley Longman Publishing Co., Inc.

Haji Mohammad Saleem, Kelly P Dillon, Susan Benesch, and Derek Ruths. 2017. A web of hate: Tackling hateful speech in online social spaces. *arXiv preprint arXiv:1709.10159.*

Helmut Schmid. 1995. Improvements in part-of-speech tagging with an application to german. In *In Proceedings of the ACL SIGDAT-Workshop*, pages 47–50.

Mike Thelwall. 2008. Social networks, gender, and friending: An analysis of myspace member profiles. *Journal of the American Society for Information Science and Technology*, 59(8):1321–1330.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26. Association for Computational Linguistics.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting hate speech on twitter using a convolution-gru based deep neural network. In *European Semantic Web Conference*, pages 745–760. Springer.

Fermi at SemEval-2019 Task 5: Using Sentence Embeddings to identify Hate Speech against Immigrants and Women on Twitter

Vijayasaradhi Indurthi[1,3], Bakhtiyar Syed[1], Manish Shrivastava[1]
Nikhil Chakravartula[3], Manish Gupta[1,2], Vasudeva Varma[1]
[1] IIIT Hyderabad, India
[2] Microsoft, India
[3] Teradata, India
[1]{vijaya.saradhi, syed.b}@research.iiit.ac.in
[1]{m.shrivastava, manish.gupta, vv}@iiit.ac.in
[2]gmanish@microsoft.com [3]nikhil.chakravartula@teradata.com

Abstract

This paper describes our system (Fermi) for Task 5 of SemEval-2019: HatEval: Multilingual Detection of Hate Speech Against Immigrants and Women on Twitter. We participated in the subtask A for English and ranked first in the evaluation on the test set. We evaluate the quality of multiple sentence embeddings and explore multiple training models to evaluate the performance of simple yet effective embedding-ML combination algorithms. Our team - **Fermi**'s model achieved an accuracy of 65.00% for English language in task A. Our models, which use pretrained Universal Encoder sentence embeddings for transforming the input and SVM (with RBF kernel) for classification, scored first position (among 68) in the leaderboard on the test set for Subtask A in English language. In this paper we provide a detailed description of the approach, as well as the results obtained in the task.

1 Introduction

Microblogging platforms like Twitter provide channels to exchange ideas using short messages called tweets. While such a platform can be used for constructive ideas, a small group of people can propagate their notions including hatred against an individual, or a group or a race to the entire world in a few seconds. This necessitates the need to come up with computational methods to identify hate speech in user generated content.

Using computational methods to identify offense, aggression and hate speech in user generated content has been gaining attention in the recent years as evidenced in (Waseem et al., 2017; Davidson et al., 2017; Malmasi and Zampieri, 2017; Kumar et al., 2018) and workshops such as Abusive Language Workshop (ALW) [1] and Workshop on Trolling, Aggression and Cyberbullying (TRAC) [2].

2 Related Work

In this section we briefly describe other work in this area.

A few of the early works related to hate speech detection employed the use of features like bag of words, word and character n-grams with relatively off-the-shelf machine learning classifiers for detection (Dinakar et al., 2011; Waseem and Hovy, 2016; Nobata et al., 2016). Deep learning methods for hate speech detection were used by Badjatiya et al. (2017) wherein the authors experimented with a combination of multiple deep learning architectures along with randomly initialized word embeddings learned by Long Short Term Memory (LSTM) models.

Papers published in the last two years include the surveys by (Schmidt and Wiegand, 2017) and (Fortuna and Nunes, 2018), the paper by (Davidson et al., 2017) which presented the Hate Speech Detection dataset used in (Malmasi and Zampieri, 2017) and a few other recent papers such as (ElSherief et al., 2018; Gambäck and Sikdar, 2017; Zhang et al., 2018).

A proposal of typology of abusive language sub-tasks is presented in (Waseem et al., 2017). For studies on languages other than English see (Su et al., 2017) on Chinese and (Fišer et al., 2017) on Slovene. Finally, for recent discussion on identifying profanity versus hate speech see (Malmasi and Zampieri, 2018). This work highlighted the challenges of distinguishing between profanity, and threatening language which may not actually contain profane language.

Some of the similar and related previous workshops are Text Analytics for Cybersecurity and

[1]https://sites.google.com/view/alw2018

[2]https://sites.google.com/view/trac1

Online Safety (TA-COS) [3], Abusive Language Workshop [4], and TRAC [5]. Related shared tasks include GermEval (Wiegand et al., 2018) and TRAC (Kumar et al., 2018).

3 Methodology

In this paper, we make use of several word embedding and sentence embedding methods.

3.1 Word Embeddings

Word embeddings have been widely used in modern Natural Language Processing applications as they provide vector representation of words. They capture the semantic properties of words and the linguistic relationship between them. These word embeddings have improved the performance of many downstream tasks across many domains like text classification, machine comprehension etc. (Camacho-Collados and Pilehvar, 2018). Multiple ways of generating word embeddings exist, such as Neural Probabilistic Language Model (Bengio et al., 2003), Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and more recently ELMo (Peters et al., 2018).

These word embeddings rely on the distributional linguistic hypothesis. They differ in the way they capture the meaning of the words or the way they are trained. Each word embedding captures a different set of semantic attributes which may or may not be captured by other word embeddings. In general, it is difficult to predict the relative performance of these word embeddings on downstream tasks. The choice of which word embeddings should be used for a given downstream task depends on experimentation and evaluation.

3.2 Sentence Embeddings

While word embeddings can produce representations for words which can capture the linguistic properties and the semantics of the words, the idea of representing sentences as vectors is an important and open research problem (Conneau et al., 2017).

Finding a universal representation of a sentence which works with a variety of downstream tasks is the major goal of many sentence embedding techniques. A common approach of obtaining a sentence representation using word embeddings is by the simple and naïve way of using the simple arithmetic mean of all the embeddings of the words present in the sentence. Smooth inverse frequency, which uses weighted averages and modifies it using Singular Value Decomposition (SVD), has been a strong contender as a baseline over traditional averaging technique (Arora et al., 2016). Other sentence embedding techniques include p-means (Rücklé et al., 2018), InferSent (Conneau et al., 2017), SkipThought (Kiros et al., 2015), Universal Encoder (Cer et al., 2018).
Task A (Hate speech detection) is a two-class classification where systems have to predict whether a tweet in English or in Spanish with a given target (women or immigrants) is hateful or not hateful. TASK B (Aggressive behavior and Target Classification) is a two-class classification where systems have to classify hateful tweets (e.g., tweets where Hate Speech against women or immigrants has been identified) as aggressive or not aggressive, and second to identify the target harassed as individual or generic (i.e. single human or group).

We formulate sub-task A of HatEval as a text classification tasks. In this paper, we evaluate various pre-trained sentence embeddings for identifying the offense, hate and aggression. We train multiple models using different machine learning algorithms to evaluate the efficacy of each of the pre-trained sentence embeddings for the downstream task. We observe that there is a class label imbalance in the dataset. To prevent any bias induced due to imbalanced classes, we process the transformed training dataset using SMOTE (Chawla et al., 2002) which synthetically oversamples data and ensures that all the classes have same number of instances.

In the following, we discuss various popular sentence embedding methods in brief.

- InferSent (Conneau et al., 2017) is a set of embeddings proposed by Facebook. InferSent embeddings have been trained using the popular language inference corpus. Given two sentences the model is trained to infer whether they are a contradiction, a neutral pairing, or an entailment. The output is an embedding of 4096 dimensions.

- Concatenated Power Mean Word Embedding (Rücklé et al., 2018) generalizes the concept of average word embeddings to power mean word embeddings. The concatenation of different types of power mean word embeddings

[3] `http://ta-cos.org/`
[4] `https://sites.google.com/site/alw2018`
[5] `https://sites.google.com/view/trac1`

Model	LR		RF		SVM-RBF		XGB	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
InferSent	64.26	64.34	63.96	62.45	57.13	41.54	**71.18**	**71.21**
Concat-p mean	63.35	63.43	67.17	65.83	63.86	60.98	71.08	70.67
Lexical Vectors	67.27	66.61	67.97	67.09	58.53	46.03	67.87	68.31
Universal Encoder	70.58	70.63	70.48	70.05	57.13	41.54	64.26	64.34
ELMo	69.68	69.78	65.96	65.12	68.37	68.44	66.57	66.59
NNLM	66.57	66.46	64.36	62.83	65.56	63.88	66.37	65.74

Table 1: *Dev* Set Accuracy and Macro-F1 scores(in percentage) for **Sub-Task A- English.**

considerably closes the gap to state-of-the-art methods mono-lingually and substantially outperforms many complex techniques cross-lingually.

- Lexical Vectors (Salle and Villavicencio, 2018) is another word embedding similar to fastText with slightly modified objective. FastText (Bojanowski et al., 2016) is another word embedding model which incorporates character n-grams into the skipgram model of Word2Vec and considers the sub-word information.

- The Universal Sentence Encoder (Cer et al., 2018) encodes text into high dimensional vectors. The model is trained and optimized for greater-than-word length text, such as sentences, phrases or short paragraphs. It is trained on a variety of data sources and a variety of tasks with the aim of dynamically accommodating a wide variety of natural language understanding tasks. The input is variable length English text and the output is a 512 dimensional vector.

- Deep Contextualized Word Representations (ELMo) (Peters et al., 2018) use language models to get the embeddings for individual words. The entire sentence or paragraph is taken into consideration while calculating these embedding representations. ELMo uses a pre-trained bi-directional LSTM language model. For the input supplied, the ELMo architecture extracts the hidden state of each layer. A weighted sum is computed of the hidden states to obtain an embedding for each sentence.

Using each of the sentence embeddings we have mentioned above, we seek to evaluate how each of them performs when the vector representations are supplied for classification with various off-the-shelf machine learning algorithms. For each of the evaluation tasks, we perform experiments using each of the sentence embeddings mentioned above and show our classification performance on the *dev* set given by the task organizers.

Using each of the sentence embeddings we have mentioned above, we seek to evaluate how each of them performs when the vector representations are supplied for classification with various off-the-shelf machine learning algorithms. For each of the evaluation tasks, we perform experiments using each of the sentence embeddings mentioned above and show our classification performance on the *dev* set given by the task organizers.

4 Dataset

The data collection methods used to compile the dataset used in HatEval is described in (Basile et al., 2019). We did not use any external datasets to augment the data for training our models.

5 Results and Analysis

The official *test* set results scored on CodaLab have been presented below in Table 2.

System	F1 (macro)	Accuracy
Universal Encoder	0.65	0.65

Table 2: Results for Sub-task A using Universal Encoder Sentence embeddings with SVM classifier using RBF kernel.

Our results on the different algorithms from the ones stated above have been mentioned henceforth and described in Table 1.

As described in Table 1 the dev set macro-averaged F-1 and accuracy is given for the task A-English.

We notice the best performance for task A in English on the official test set was bagged by the model which used pretrained Universal sentence embeddings using SVM with RBF kernel. However, pretrained Infersent embeddings along with XGBoost algorithm outperformed every other combination on the dev test. This can be probably due to the difference between the distributions in the dev and the official test sets.

Overall, this work shows how different set of pretrained embeddings trained from different state-of-the-art architectures and methods when used with simple machine learning classifiers perform very well in the classification task of categorizing text as offensive or not.

6 Conclusions and Future Work

It is also important to note that the experiments are performed using the default parameters, so there is much scope for improvement with a lot of fine-tuning, which we plan on considering for future research purposes. Further, we can explore augmenting data from other similar shared tasks to achieve better performance.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Jose Camacho-Collados and Mohammad Taher Pilehvar. 2018. From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research*, 63:743–788.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Eleventh International AAAI Conference on Web and Social Media*.

Karthik Dinakar, Roi Reichart, and Henry Lieberman. 2011. Modeling the detection of textual cyberbullying. In *The Social Mobile Web*, pages 11–17.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Darja Fišer, Tomaž Erjavec, and Nikola Ljubešić. 2017. Legal Framework, Dataset and Annotation Schema for Socially Unacceptable On-line Discourse Practices in Slovene. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 467–472.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive Language Detection in Online User Content. In *Proceedings of the 25th International Conference on World Wide Web*, pages 145–153. International World Wide Web Conferences Steering Committee.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Andreas Rücklé, Steffen Eger, Maxime Peyrard, and Iryna Gurevych. 2018. Concatenated p-mean word embeddings as universal cross-lingual sentence representations. *arXiv preprint arXiv:1803.01400*.

Alexandre Salle and Aline Villavicencio. 2018. Incorporating subword information into matrix factorization word embeddings. *arXiv preprint arXiv:1805.03710*.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Huei-Po Su, Chen-Jie Huang, Hao-Tsung Chang, and Chuan-Jie Lin. 2017. Rephrasing Profanity in Chinese Text. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *SRW@HLT-NAACL*.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval)

Marcos Zampieri,[1] Shervin Malmasi,[2] Preslav Nakov,[3]
Sara Rosenthal,[4] Noura Farra,[5] Ritesh Kumar[6]
[1]University of Wolverhampton, UK, [2]Amazon Research, USA
[3]Qatar Computing Research Institute, HBKU, Qatar
[4]IBM Research, USA, [5]Columbia University, USA, [6]Bhim Rao Ambedkar University, India
m.zampieri@wlv.ac.uk

Abstract

We present the results and the main findings of SemEval-2019 Task 6 on Identifying and Categorizing Offensive Language in Social Media (OffensEval). The task was based on a new dataset, the Offensive Language Identification Dataset (OLID), which contains over 14,000 English tweets. It featured three sub-tasks. In sub-task A, the goal was to discriminate between offensive and non-offensive posts. In sub-task B, the focus was on the type of offensive content in the post. Finally, in sub-task C, systems had to detect the target of the offensive posts. OffensEval attracted a large number of participants and it was one of the most popular tasks in SemEval-2019. In total, about 800 teams signed up to participate in the task, and 115 of them submitted results, which we present and analyze in this report.

1 Introduction

Recent years have seen the proliferation of offensive language in social media platforms such as Facebook and Twitter. As manual filtering is very time consuming, and as it can cause post-traumatic stress disorder-like symptoms to human annotators, there have been many research efforts aiming at automating the process. The task is usually modeled as a supervised classification problem, where systems are trained on posts annotated with respect to the presence of some form of abusive or offensive content. Examples of offensive content studied in previous work include hate speech (Davidson et al., 2017; Malmasi and Zampieri, 2017, 2018), cyberbulling (Dinakar et al., 2011), and aggression (Kumar et al., 2018). Moreover, given the multitude of terms and definitions used in the literature, some recent studies have investigated the common aspects of different abusive language detection sub-tasks (Waseem et al., 2017; Wiegand et al., 2018).

Interestingly, none of this previous work has studied both the type *and* the target of the offensive language, which is our approach here. Our task, OffensEval[1], uses the Offensive Language Identification Dataset (OLID)[2] (Zampieri et al., 2019), which we created specifically for this task. OLID is annotated following a hierarchical three-level annotation schema that takes both the target and the type of offensive content into account. Thus, it can relate to phenomena captured by previous datasets such as the one by Davidson et al. (2017). Hate speech, for example, is commonly understood as an insult targeted at a *group*, whereas cyberbulling is typically targeted at an *individual*.

We defined three sub-tasks, corresponding to the three levels in our annotation schema:[3]

Sub-task A: Offensive language identification (104 participating teams)

Sub-task B: Automatic categorization of offense types (71 participating teams)

Sub-task C: Offense target identification (66 participating teams)

The remainder of this paper is organized as follows: Section 2 discusses prior work, including shared tasks related to OffensEval. Section 3 presents the shared task description and the subtasks included in OffensEval. Section 4 includes a brief description of OLID based on (Zampieri et al., 2019). Section 5 discusses the participating systems and their results in the shared task. Finally, Section 6 concludes and suggests directions for future work.

[1]http://competitions.codalab.org/competitions/20011
[2]http://scholar.harvard.edu/malmasi/olid
[3]A total of 800 teams signed up to participate in the task, but only 115 teams ended up submitting results eventually.

2 Related Work

Different abusive and offense language identification problems have been explored in the literature ranging from aggression to cyber bullying, hate speech, toxic comments, and offensive language. Below we discuss each of them briefly.

Aggression identification: The TRAC shared task on Aggression Identification (Kumar et al., 2018) provided participants with a dataset containing 15,000 annotated Facebook posts and comments in English and Hindi for training and validation. For testing, two different sets, one from Facebook and one from Twitter, were used. The goal was to discriminate between three classes: non-aggressive, covertly aggressive, and overtly aggressive. The best-performing systems in this competition used deep learning approaches based on convolutional neural networks (CNN), recurrent neural networks, and LSTM (Aroyehun and Gelbukh, 2018; Majumder et al., 2018).

Bullying detection: There have been several studies on cyber bullying detection. For example, Xu et al. (2012) used sentiment analysis and topic models to identify relevant topics, and Dadvar et al. (2013) used user-related features such as the frequency of profanity in previous messages.

Hate speech identification: This is the most studied abusive language detection task (Kwok and Wang, 2013; Burnap and Williams, 2015; Djuric et al., 2015). More recently, Davidson et al. (2017) presented the hate speech detection dataset with over 24,000 English tweets labeled as non offensive, hate speech, and profanity.

Offensive language: The GermEval[4] (Wiegand et al., 2018) shared task focused on offensive language identification in German tweets. A dataset of over 8,500 annotated tweets was provided for a course-grained binary classification task in which systems were trained to discriminate between offensive and non-offensive tweets. There was also a second task where the offensive class was subdivided into profanity, insult, and abuse. This is similar to our work, but there are three key differences: (*i*) we have a third level in our hierarchy, (*ii*) we use different labels in the second level, and (*iii*) we focus on English.

Toxic comments: The Toxic Comment Classification Challenge[5] was an open competition at Kaggle, which provided participants with comments from Wikipedia organized in six classes: toxic, severe toxic, obscene, threat, insult, identity hate. The dataset was also used outside of the competition (Georgakopoulos et al., 2018), including as additional training material for the aforementioned TRAC shared (Fortuna et al., 2018).

While each of the above tasks tackles a particular type of abuse or offense, there are many commonalities. For example, an insult targeted at an individual is commonly known as cyberbulling and insults targeted at a group are known as hate speech. The hierarchical annotation model proposed in OLID (Zampieri et al., 2019) and used in OffensEval aims to capture this. We hope that the OLID's dataset would become a useful resource for various offensive language identification tasks.

3 Task Description and Evaluation

The training and testing material for OffensEval is the aforementioned Offensive Language Identification Dataset (OLID) dataset, which was built specifically for this task. OLID was annotated using a hierarchical three-level annotation model introduced in Zampieri et al. (2019). Four examples of annotated instances from the dataset are presented in Table 1. We use the annotation of each of the three layers in OLID for a sub-task in OffensEval as described below.

3.1 Sub-task A: Offensive language identification

In this sub-task, the goal is to discriminate between offensive and non-offensive posts. Offensive posts include insults, threats, and posts containing any form of untargeted profanity. Each instance is assigned one of the following two labels.

- Not Offensive (NOT): Posts that do not contain offense or profanity;
- Offensive (OFF): We label a post as offensive if it contains any form of non-acceptable language (profanity) or a targeted offense, which can be veiled or direct. This category includes insults, threats, and posts containing profane language or swear words.

[4] `http://projects.fzai.h-da.de/iggsa/`

[5] `http://kaggle.com/c/jigsaw-toxic-comment-classification-challenge`

Tweet	A	B	C
@USER He is so generous with his offers.	NOT	—	—
IM FREEEEE!!!! WORST EXPERIENCE OF MY FUCKING LIFE	OFF	UNT	—
@USER Fuk this fat cock sucker	OFF	TIN	IND
@USER Figures! What is wrong with these idiots? Thank God for @USER	OFF	TIN	GRP

Table 1: Four tweets from the OLID dataset, with their labels for each level of the annotation model.

3.2 Sub-task B: Automatic categorization of offense types

In sub-task B, the goal is to predict the type of offense. Only posts labeled as Offensive (OFF) in sub-task A are included in sub-task B. The two categories in sub-task B are the following:

- Targeted Insult (TIN): Posts containing an insult/threat to an individual, group, or others (see sub-task C below);
- Untargeted (UNT): Posts containing non-targeted profanity and swearing. Posts with general profanity are not targeted, but they contain non-acceptable language.

3.3 Sub-task C: Offense target identification

Sub-task C focuses on the target of offenses. Only posts that are either insults or threats (TIN) arwe considered in this third layer of annotation. The three labels in sub-task C are the following:

- Individual (IND): Posts targeting an individual. It can be a a famous person, a named individual or an unnamed participant in the conversation. Insults/threats targeted at individuals are often defined as cyberbullying.
- Group (GRP): The target of these offensive posts is a group of people considered as a unity due to the same ethnicity, gender or sexual orientation, political affiliation, religious belief, or other common characteristic. Many of the insults and threats targeted at a group correspond to what is commonly understood as hate speech.
- Other (OTH): The target of these offensive posts does not belong to any of the previous two categories, e.g., an organization, a situation, an event, or an issue.

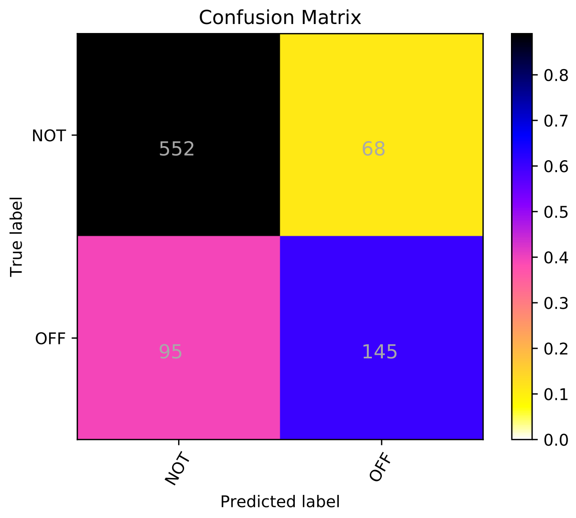


Figure 1: Example of a confusion matrix provided in the results package for team NULI, which is the best-performing team for sub-task A.

3.4 Task Evaluation

Given the strong imbalance between the number of instances in the different classes across the three tasks, we used the macro-averaged F1-score as the official evaluation measure for all three sub-tasks.

At the end of the competition, we provided the participants with packages containing the results for each of their submissions, including tables and confusion matrices, and tables with the ranks listing all teams who competed in each sub-task. For example, the confusion matrix for the best team in sub-task A is shown in Figure 1.

3.5 Participation

The task attracted nearly 800 teams and 115 of them submitted their results. The teams that submitted papers for the SemEval-2019 proceedings are listed in Table 2.[6]

[6] *ASE-CSE* is for *Amrita School of Engineering - CSE.*

Team	System Description Paper
Amobee	(Rozental and Biton, 2019)
ASE-CSE	(Sridharan and T, 2019)
bhanodaig	(Kumar et al., 2019)
BNU-HKBU ...	(Wu et al., 2019)
CAMsterdam	(Aglionby et al., 2019)
CN-HIT-MI.T	(Yaojie et al., 2019)
ConvAI	(Pavlopoulos et al., 2019)
DA-LD-Hildesheim	(Modha et al., 2019)
DeepAnalyzer	(la Pea and Rosso, 2019)
Duluth	(Pedersen, 2019)
Emad	(Kebriaei et al., 2019)
Embeddia	(Pelicon et al., 2019)
Fermi	(Indurthi et al., 2019)
Ghmerti	(Doostmohammadi et al., 2019)
HAD-Tübingen	(Bansal et al., 2019)
HHU	(Oberstrass et al., 2019)
Hope	(Patras et al., 2019)
INGEOTEC	(Graff et al., 2019)
JCTICOL	(HaCohen-Kerner et al., 2019)
jhan014	(Han et al., 2019)
JTML	(Torres and Vaca, 2019)
JU_ETCE_17_21	(Mukherjee et al., 2019)
KMI_Coling	(Rani and Ojha, 2019)
LaSTUS/TALN	(Altin et al., 2019)
LTL-UDE	(Aggarwal et al., 2019)
MIDAS	(Mahata et al., 2019)
Nikolov-Radivchev	(Nikolov and Radivchev, 2019)
NIT_Agartala_NLP_Team	(Swamy et al., 2019)
NLP	(Kapil et al., 2019)
NLP@UIOWA	(Rusert and Srinivasan, 2019)
NLPR@SRPOL	(Seganti et al., 2019)
nlpUP	(Mitrović et al., 2019)
NULI	(Liu et al., 2019)
SINAI	(Plaza-del Arco et al., 2019)
SSN_NLP	(Thenmozhi et al., 2019)
Stop PropagHate	(Fortuna et al., 2019)
Pardeep	(Singh and Chand, 2019)
techssn	(S et al., 2019)
The Titans	(Garain and Basu, 2019)
TUVD	(Shushkevich et al., 2019)
TüKaSt	(Kannan and Stein, 2019)
UBC-NLP	(Rajendran et al., 2019)
UTFPR	(Paetzold, 2019)
UHH-LT	(Wiedemann et al., 2019)
UM-IU@LING	(Zhu et al., 2019)
USF	(Goel and Sharma, 2019)
UVA Wahoos	(Ramakrishnan et al., 2019)
YNU-HPCC	(Zhou et al., 2019)
YNUWB	(Wang et al., 2019)
Zeyad	(El-Zanaty, 2019)

Table 2: The teams that participated in OffensEval and submitted system description papers.

4 Data

Below, we briefly describe OLID, the dataset used for our SemEval-2019 task 6. A detailed description of the data collection process and annotation is presented in Zampieri et al. (2019).

OLID is a large collection of English tweets annotated using a hierarchical three-layer annotation model. It contains 14,100 annotated tweets divided into a training partition of 13,240 tweets and a testing partition of 860 tweets. Additionally, a small trial dataset of 320 tweets was made available before the start of the competition.

A	B	C	Train	Test	Total
OFF	TIN	IND	2,407	100	2,507
OFF	TIN	OTH	395	35	430
OFF	TIN	GRP	1,074	78	1,152
OFF	UNT	—	524	27	551
NOT	—	—	8,840	620	9,460
All			13,240	860	14,100

Table 3: Distribution of label combinations in OLID.

The distribution of the labels in OLID is shown in Table 3. We annotated the dataset using the crowdsourcing platform Figure Eight.[7] We ensured the quality of the annotation by only hiring experienced annotators on the platform and by using test questions to discard annotators who did not achieve a certain threshold. All the tweets were annotated by two people. In case of disagreement, a third annotation was requested, and ultimately we used a majority vote. Examples of tweets from the dataset with their annotation labels are shown in Table 1.

5 Results

The models used in the task submissions ranged from traditional machine learning, e.g., SVM and logistic regression, to deep learning, e.g., CNN, RNN, BiLSTM, including attention mechanism, to state-of-the-art deep learning models such as ELMo (Peters et al., 2018) and BERT (Devlin et al.). Figure 2 shows a pie chart indicating the breakdown by model type for all participating systems in sub-task A. Deep learning was clearly the most popular approach, as were also ensemble models. Similar trends were observed for sub-tasks B and C.

[7]https://www.figure-eight.com/

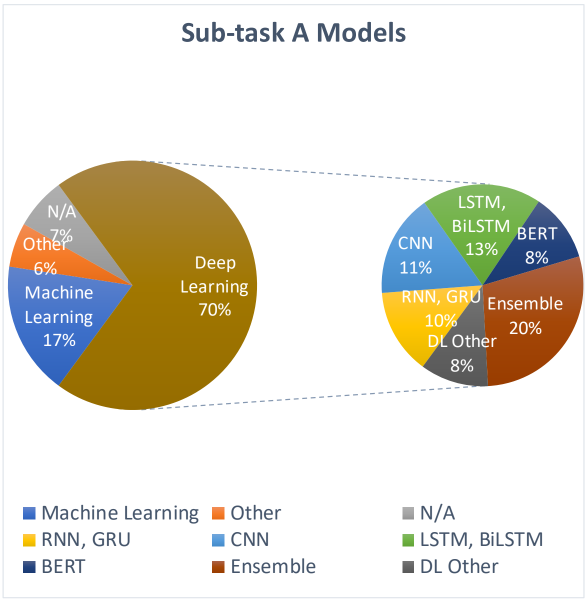


Figure 2: Pie chart showing the models used in sub-task A. 'N/A' indicates that the system did not have a description.

Some teams used additional training data, exploring external datasets such as Hate Speech Tweets (Davidson et al., 2017), toxicity labels (Thain et al., 2017), and TRAC (Kumar et al., 2018). Moreover, seven teams indicated that they used sentiment lexicons or a sentiment analysis model for prediction, and two teams reported the use of offensive word lists. Furthermore, several teams used pre-trained word embeddings from FastText (Bojanowski et al., 2016), from GloVe, including Twitter embeddings from GloVe (Pennington et al., 2014) and from word2vec (Mikolov et al., 2013; Godin et al., 2015).

In addition, several teams used techniques for pre-processing the tweets such as normalizing the tokens, hashtags, URLs, retweets (RT), dates, elongated words (e.g., "Hiiiii" to "Hi", partially hidden words ("c00l" to "cool"). Other techniques include converting emojis to text, removing uncommon words, and using Twitter-specific tokenizers, such as the Ark Tokenizer[8] (Gimpel et al., 2011) and the NLTK TweetTokenizer,[9] as well as standard tokenizers (Stanford Core NLP (Manning et al., 2014), and the one from Keras.[10] Approximately a third of the teams indicated that they used one or more of these techniques.

[8] http://www.cs.cmu.edu/~ark/TweetNLP

[9] http://www.nltk.org/api/nltk.tokenize.html

[10] http://keras.io/preprocessing/text/

The results for each of the sub-tasks are shown in Table 4. Due to the large number of submissions, we only show the F1-score for the top-10 teams, followed by result ranges for the rest of the teams. We further include the models and the baselines from (Zampieri et al., 2019): CNN, BiLSTM, and SVM. The baselines are choosing all predictions to be of the same class, e.g., all offensive, and all not offensive for sub-task A. Table 5 shows all the teams that participated in the tasks along with their ranks in each task. These two tables can be used together to find the score/range for a particular team.

Below, we describe the overall results for each sub-task, and we describe the top-3 systems.

5.1 Sub-task A

Sub-task A was the most popular sub-task with 104 participating teams. Among the top-10 teams, seven used BERT (Devlin et al.) with variations in the parameters and in the pre-processing steps. The top-performing team, *NULI*, used BERT-base-uncased with default-parameters, but with a max sentence length of 64 and trained for 2 epochs. The 82.9% F1 score of NULI is 1.4 points better than the next system, but the difference between the next 5 systems, ranked 2-6, is less than one point: 81.5%-80.6%. The top non-BERT model, *MIDAS*, is ranked sixth. They used an ensemble of CNN and BLSTM+BGRU, together with Twitter word2vec embeddings (Godin et al., 2015) and token/hashtag normalization.

5.2 Sub-task B

A total of 76 teams participated in sub-task B, and 71 of them had also participated in sub-task A. In contrast to sub-task A, where BERT clearly dominated, here five of the top-10 teams used an ensemble model. Interestingly, the best team, *jhan014*, which was ranked 76th in sub-task A, used a rule-based approach with a keyword filter based on a Twitter language behavior list, which included strings such as hashtags, signs, etc., achieving an F1-score of 75.5%. The second and the third teams, *Amobee* and *HHU*, used ensembles of deep learning (including BERT) and non-neural machine learning models. The best team from sub-task A also performed well here, ranked 4th (71.6%), thus indicating that overall BERT works well for sub-task B as well.

Sub-task A		Sub-task B		Sub-task C	
Team Ranks	**F1 Range**	**Team Ranks**	**F1 Range**	**Team Ranks**	**F1 Range**
1	0.829	1	0.755	1	0.660
2	0.815	2	0.739	2	0.628
3	0.814	3	0.719	3	0.626
4	0.808	4	0.716	4	0.621
5	0.807	5	0.708	5	0.613
6	0.806	6	0.706	6	0.613
7	0.804	7	0.700	7	0.591
8	0.803	8	0.695	8	0.588
9	0.802	9	0.692	9	0.587
CNN	**0.800**	**CNN**	**0.690**	10	0.586
10	0.798	10	0.687	11-14	.571-.580
11-12	.793-.794	11-14	.680-.682	15-18	.560-.569
13-23	.782-.789	15-24	.660-.671	19-23	.547-.557
24-27	.772-.779	**BiLSTM**	**0.660**	24-29	.523-.535
28-31	.765-.768	25-29	.640-.655	30-33	.511-.515
32-40	.750-.759	**SVM**	**0.640**	34-40	.500-.509
BiLSTM	**0.750**	30-38	.600-.638	41-47	.480-.490
41-45	.740-.749	39-49	.553-.595	**CNN**	**0.470**
46-57	.730-.739	50-62	.500-.546	**BiLSTM**	**0.470**
58-63	.721-.729	**ALL TIN**	**0.470**	**SVM**	**0.450**
64-71	.713-.719	63-74	.418-.486	46-60	.401-.476
72-74	.704-.709	75	0.270	61-65	.249-.340
SVM	**0.690**	76	0.121	**All IND**	**0.210**
75-89	.619-.699	**All UNT**	**0.100**	**All GRP**	**0.180**
90-96	.500-.590			**ALL OTH**	**0.090**
97-103	.422-.492				
All NOT	**0.420**				
All OFF	**0.220**				
104	0.171				

Table 4: F1-Macro for the top-10 teams followed by the rest of the teams grouped in ranges for all three sub-tasks. Refer to Table 5 to see the team names associated with each rank. We also include the models (**CNN, BiLSTM, and SVM**) and the baselines (**All NOT and All OFF**) from (Zampieri et al., 2019), shown in bold.

5.3 Sub-task C

A total of 66 teams participated in sub-task C, and most of them also participated in sub-tasks A and B. As in sub-task B, ensembles were quite successful and were used by five of the top-10 teams. However, as in sub-task A, the best team, *vradivchev_anikolov*, used BERT after trying many other deep learning methods. They also used pre-processing and pre-trained word embeddings based on GloVe. The second best team, *NLPR@SRPOL*, used an ensemble of deep learning models such as OpenAI Finetune, LSTM, Transformer, and non-neural machine learning models such as SVM and Random Forest.

5.4 Description of the Top Teams

The top-3 teams by average rank for all three sub-tasks were *NLPR@SRPOL*, *NULI*, and *vradivchev_anikolov*. Below, we provide a brief description of their approaches:

NLPR@SRPOL was ranked 8th, 9th, and 2nd on sub-tasks A, B, and C, respectively. They used ensembles of OpenAI GPT, Random Forest, the Transformer, Universal encoder, ELMo, and combined embeddings from fastText and custom ones. They trained their models on multiple publicly available offensive datasets, as well as on their own custom dataset annotated by linguists.

	Sub-task				**Sub-task**				**Sub-task**		
Team	**A**	**B**	**C**	**Team**	**A**	**B**	**C**	**Team**	**A**	**B**	**C**
NULI	**1**	4	18	resham	40	43	-	kroniker	79	71	-
<u>**vradivchev_anikolov**</u>	<u>2</u>	16	**1**	Xcosmos	41	47	29	aswathyprem	80	-	-
UM-IU@LING	3	76	27	jkolis	42	-	-	DeepAnalyzer	81	38	45
Embeddia	4	18	5	NIT_Agartala_NLP_Team	43	5	38	Code Lyoko	82	-	-
MIDAS	5	8	-	Stop PropagHate	44	-	-	rowantahseen	83	-	-
BNU-HKBU	6	62	39	KVETHZ	45	52	26	ramjib	84	-	-
SentiBERT	7	-	-	christoph.alt	46	14	36	OmerElshrief	85	-	-
<u>NLPR@SRPOL</u>	8	9	<u>2</u>	TECHSSN	47	22	16	desi	86	56	-
YNUWB	9	-	-	USF	48	32	62	Fermi	87	31	3
LTL-UDE	10	-	19	Ziv_Ben_David	49	64	33	mkannan	88	-	-
nlpUP	11	-	-	JCTICOL	50	63	-	mking	89	35	54
ConvAI	12	11	35	TüKaSt	51	23	50	ninab	90	69	-
Vadym	13	10	-	Gal_DD	52	66	25	dianalungu725	91	74	65
UHH-LT	14	21	13	HAD-Tübingen	53	59	61	Halamulki	92	-	-
CAMsterdam	15	19	20	Emad	54	-	-	SSN_NLP	93	65	64
YNU-HPCC	16	-	-	NLP@UIOWA	55	27	37	UTFPR	94	-	-
nishnik	17	-	-	INGEOTEC	56	15	12	rogersdepelle	95	-	-
<u>Amobee</u>	18	<u>2</u>	7	Duluth	57	39	44	Amimul Ihsan	96	-	-
himanisoni	19	46	11	Zeyad	58	34	34	supriyamandal	97	75	-
samsam	20	-	-	ShalomRochman	59	70	58	ramitpahwa	98	-	-
JU_ETCE_17_21	21	50	47	stefaniehegele	60	-	-	ASE - CSE	99	33	32
DA-LD-Hildesheim	22	28	21	NLP-CIC	61	48	46	kripo	100	-	-
YNU-HPCC	23	12	4	Elyash	62	67	40	garain	101	44	63
ChenXiuling	24	-	28	KMI_Coling	63	45	53	NAYEL	102	-	-
Ghmerti	25	29	-	RUG_OffenseEval	64	-	-	magnito60	103	-	-
safina	26	-	-	jaypee1996	65	41	-	AyushS	104	36	48
Arjun Roy	27	17	-	orabia	66	55	8	UBC_NLP	-	6	9
CN-HIT-MI.T	28	30	22	v.gambhir15	67	58	60	bhanodaig	-	57	-
LaSTUS/TALN	29	20	15	kerner-jct.ac.il	68	68	42	Panaetius	-	60	-
HHU	30	3	-	SINAI	69	-	-	eruppert	-	61	-
na14	31	26	10	apalmer	70	13	55	Macporal	-	72	-
NRC	32	37	24	ayman	71	53	57	NoOffense	-	-	6
NLP	33	54	52	Geetika	72	24	-	HHU	-	-	14
JTML	34	-	-	Taha	73	51	59	quanzhi	-	-	17
Arup-Baruah	35	25	31	justhalf	74	-	-	TUVD	-	-	23
UVA_Wahoos	36	42	-	Pardeep	75	7	41	mmfouad	-	-	51
NLP@UniBuc	37	73	49	**jhan014**	76	**1**	30	balangheorghe	-	-	56
NTUA-ISLab	38	40	43	liuxy94	77	-	-				
Rohit	39	49	-	ngre1989	78	-	-				

Table 5: All the teams that participated in SemEval-2019 Task 6 with their ranks for each sub-task. The symbol '-' indicates that the team did not participate in some of the subtasks. Please, refer to Table 4 to see the scores based on a team's rank. The top team for each task is in **bold**, and the second-place team is <u>underlined</u>. Note: *ASE - CSE* stands for *Amrita School of Engineering - CSE*, and *BNU-HBKU* stands for *BNU-HKBU UIC NLP Team 2*.

NULI was ranked 1st, 4th, and 18th on sub-tasks A, B, and C, respectively. They experimented with different models including linear models, LSTM, and pre-trained BERT with fine-tuning on the OLID dataset. Their final submissions for all three subtasks only used BERT, which performed best during development. They also used a number of pre-processing techniques such as hashtag segmentation and emoji substitution.

vradivchev_anikolov was ranked 2nd, 16th, and 1st on sub-tasks A, B, and C, respectively. They trained a variety of models and combined them in ensembles, but their best submissions for sub-tasks A and C used BERT only, as the other models overfitted. For sub-task B, BERT did not perform as well, and they used soft voting classifiers. In all cases, they used pre-trained GloVe vectors and they also applied techniques to address the class imbalance in the training data.

6 Conclusion

We have described SemEval-2019 Task 6 on Identifying and Categorizing Offensive Language in Social Media (OffensEval). The task used OLID (Zampieri et al., 2019), a dataset of English tweets annotated for offensive language use, following a three-level hierarchical schema that considers (*i*) whether a message is offensive or not (for sub-task A), (*ii*) what is the type of the offensive message (for sub-task B), and (*iii*) who is the target of the offensive message (for sub-task C).

Overall, about 800 teams signed up for OffensEval, and 115 of them actually participated in at least one sub-task. The evaluation results have shown that the best systems used ensembles and state-of-the-art deep learning models such as BERT. Overall, both deep learning and traditional machine learning classifiers were widely used. More details about the indvididual systems can be found in their respective system description papers, which are published in the SemEval-2019 proceedings. A list with references to these publications can be found in Table 2; note, however, that only 50 of the 115 participating teams submitted a system description paper.

As is traditional for SemEval, we have made OLID publicly available to the research community beyond the SemEval competition, hoping to facilitate future research on this important topic. In fact, the OLID dataset and the SemEval-2019 Task 6 competition setup have already been used in teaching curricula in universities in UK and USA. For example, student competitions based on OffensEval using OLID have been organized as part of Natural Language Processing and Text Analytics courses in two universities in UK: Imperial College London and the University of Leeds. System papers describing some of the students' work are publicly accessible[11] and have also been made available on arXiv.org (Cambray and Podsadowski, 2019; Frisiani et al., 2019; Ong, 2019; Sapora et al., 2019; Puiu and Brabete, 2019; Uglow et al., 2019). Similarly, a number of students in Linguistics and Computer Science at the University of Arizona in USA have been using OLID in their coursework.

In future work, we plan to increase the size of the OLID dataset, while addressing issues such as class imbalance and the small size for the test partition, particularly for sub-tasks B and C. We would also like to expand the dataset and the task to other languages.

Acknowledgments

We would like to thank the SemEval-2019 organizers for hosting the OffensEval task and for replying promptly to all our inquires. We further thank the SemEval-2019 anonymous reviewers for the helpful suggestions and for the constructive feedback, which have helped us improve the text of this report.

We especially thank the SemEval-2019 Task 6 participants for their interest in the shared task, for their participation, and for their timely feedback, which have helped us make the shared task a success.

Finally, we would like to thank Lucia Specia from Imperial College London and Eric Atwell from the University of Leeds for hosting the OffensEval competition in their courses. We further thank the students who participated in these student competitions and especially those who wrote papers describing their systems.

The research presented in this paper was partially supported by an ERAS fellowship, which was awarded to Marcos Zampieri by the University of Wolverhampton, UK.

[11] `http://scholar.harvard.edu/malmasi/offenseval-student-systems`

References

Piush Aggarwal, Tobias Horsmann, Michael Wojatzki, and Torsten Zesch. 2019. LTL-UDE at SemEval-2019 Task 6: BERT and two-vote classification for categorizing offensiveness. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Guy Aglionby, Chris Davis, Pushkar Mishra, Andrew Caines, Helen Yannakoudakis, Marek Rei, Ekaterina Shutova, and Paula Buttery. 2019. CAMsterdam at SemEval-2019 Task 6: Neural and graph-based feature extraction for the identification of offensive tweets. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Lutfiye Seda Mut Altin, Alex Bravo Serrano, and Horacio Saggion. 2019. LaSTUS/TALN at SemEval-2019 Task 6: Identification and categorization of offensive language in social media with attention-based Bi-LSTM model. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Flor Miriam Plaza-del Arco, Dolores Molina-González, Teresa Martín-Valdivia, and Alfonso Ureña-López. 2019. SINAI at SemEval-2019 Task 6: Incorporating lexicon knowledge into SVM learning to identify and categorize offensive language in social media. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Segun Taofeek Aroyehun and Alexander Gelbukh. 2018. Aggression detection in social media: Using deep neural networks, data augmentation, and pseudo labeling. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC)*, pages 90–97.

Himanshu Bansal, Daniel Nagel, and Anita Soloveva. 2019. HAD-Tübingen at SemEval-2019 Task 6: Deep learning analysis of offensive language on Twitter: Identification and categorization. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR*, abs/1607.04606.

Pete Burnap and Matthew L Williams. 2015. Cyber hate speech on Twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet*, 7(2):223–242.

Aleix Cambray and Norbert Podsadowski. 2019. Bidirectional recurrent models for offensive tweet classification. *arXiv preprint arXiv:1903.08808*.

Maral Dadvar, Dolf Trieschnigg, Roeland Ordelman, and Franciska de Jong. 2013. Improving cyberbullying detection with user context. In *Advances in Information Retrieval*, pages 693–696. Springer.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technology (NAACL-HLT)*.

Karthik Dinakar, Roi Reichart, and Henry Lieberman. 2011. Modeling the detection of textual cyberbullying. In *The Social Mobile Web*, pages 11–17.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the Web Conference (WWW)*.

Ehsan Doostmohammadi, Hossein Sameti, and Ali Saffar. 2019. Ghmerti at SemEval-2019 Task 6: A deep word- and character-based approach to offensive language identification. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Zeyad El-Zanaty. 2019. Zeyad at SemEval-2019 Task 6: That's offensive! An all-out search for an ensemble to identify and categorize offense in tweets. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Paula Fortuna, José Ferreira, Luiz Pires, Guilherme Routar, and Sérgio Nunes. 2018. Merging datasets for aggressive text identification. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC)*, pages 128–139.

Paula Fortuna, Juan Soler-Company, and Nunes Srgio. 2019. Stop PropagHate at SemEval-2019 Tasks 5 and 6: Are abusive language classification results reproducible? In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Nicolò Frisiani, Alexis Laignelet, and Batuhan Güler. 2019. Combination of multiple deep learning architectures for offensive language detection in tweets. *arXiv preprint arXiv:1903.08734*.

Avishek Garain and Arpan Basu. 2019. The Titans at SemEval-2019 Task 6: Hate speech and target detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Spiros V Georgakopoulos, Sotiris K Tasoulis, Aristidis G Vrahatis, and Vassilis P Plagianakos. 2018. Convolutional neural networks for toxic comment classification. *arXiv preprint arXiv:1802.09957*.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for Twitter: annotation, features, and experiments. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Fréderic Godin, Baptist Vandersmissen, Wesley De Neve, and Rik Van de Walle. 2015. Multimedia Lab @ACL WNUT NER Shared Task: Named entity recognition for Twitter microposts using distributed word representations. In *Proceedings of the Workshop on Noisy User-generated Text*.

Bharti Goel and Ravi Sharma. 2019. USF at SemEval-2019 Task 6: Offensive language detection using LSTM with word embeddings. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Mario Graff, Sabino Miranda-Jiménez, Eric S. Tellez, and Daniela Moctezuma. 2019. INGEOTEC at SemEval-2019 Task 5 and Task 6: A genetic programming approach for text classification. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Yaakov HaCohen-Kerner, Ziv Ben-David, Gal Didi, Eli Cahn, Shalom Rochman, and Elyashiv Shayovitz. 2019. JCTICOL at SemEval-2019 Task 6: Classifying offensive language in social media using deep learning methods, word/character n-gram features, and preprocessing methods. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Jiahui Han, Xinyu Liu, and Shengtan Wu. 2019. jhan014 at SemEval-2019 Task 6: Identifying and categorizing offensive language in social media. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Vijayasaradhi Indurthi, Bakhtiyar Syed, Manish Shrivastava, Manish Gupta, and Vasudeva Varma. 2019. Fermi at SemEval-2019 Task 6: Identifying and categorizing offensive language in social media using sentence embeddings. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Madeeswaran Kannan and Lukas Stein. 2019. TüKaSt at SemEval-2019 Task 6: something old, something neu(ral): Traditional and neural approaches to offensive text classification. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Prashant Kapil, Asif Ekbal, and Dipankar Das. 2019. NLP at SemEval-2019 Task 6: Detecting offensive language using neural networks. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Emad Kebriaei, Samaneh Karimi, Nazanin Sabri, and Azadeh Shakery. 2019. Emad at SemEval-2019 Task 6: Offensive language identification using traditional machine learning and deep learning approaches. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ritesh Kumar, Guggilla Bhanodai, Rajendra Pamula, and Chennuru Maheshwar Reddy. 2019. bhanodaig at SemEval-2019 Task 6: Categorizing offensive language in social media. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ritesh Kumar, Atul Kr Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC)*.

Irene Kwok and Yuzhou Wang. 2013. Locate the hate: Detecting tweets against blacks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Ping Liu, Wen Li, and Liang Zou. 2019. NULI at SemEval-2019 Task 6: Transfer learning for offensive language detection using bidirectional transformers. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Debanjan Mahata, Haimin Zhang, Karan Uppal, Yaman Kumar, Rajiv Ratn Shah, Simra Shahid, Laiba Mehnaz, and Sarthak Anand. 2019. MIDAS at SemEval-2019 Task 6: Identifying offensive posts and targeted offense from Twitter. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Prasenjit Majumder, Thomas Mandl, et al. 2018. Filtering aggression from the multilingual social media feed. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC)*, pages 199–207.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of the Conference on Recent Advances in Natural Language Processing (RANLP)*.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1 – 16.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*.

Jelena Mitrović, Bastian Birkeneder, and Michael Granitzer. 2019. nlpUP at SemEval-2019 Task 6: a deep neural language model for offensive language detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Sandip Modha, Prasenjit Majumder, and Daksh Patel. 2019. DA-LD-Hildesheim at SemEval-2019 Task 6: Tracking offensive content with deep learning model using shallow representation. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Preeti Mukherjee, Mainak Pal, Somnath Banerjee, and Sudip Kumar Naskar. 2019. JU_ETCE_17_21 at SemEval-2019 Task 6: Efficient machine learning and neural network approaches for identifying and categorizing offensive language in tweets. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Alex Nikolov and Victor Radivchev. 2019. Nikolov-Radivchev at SemEval-2019 Task 6: Offensive tweet classification with BERT and ensembles. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Alexander Oberstrass, Julia Romberg, Anke Stoll, and Stefan Conrad. 2019. HHU at SemEval-2019 Task 6: Context does matter - tackling offensive language identification and categorization with ELMo. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ryan Ong. 2019. Offensive language analysis using deep learning architecture. *arXiv preprint arXiv:1903.05280*.

Gustavo Henrique Paetzold. 2019. UTFPR at SemEval-2019 Task 6: Relying on compositionality to find offense. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval)*.

Gabriel Florentin Patras, Diana Florina Lungu, Daniela Gifu, and Diana Trandabat. 2019. Hope at SemEval-2019 Task 6: Mining social media language to discover offensive language. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

John Pavlopoulos, Nithum Thain, Lucas Dixon, and Ion Androutsopoulos. 2019. ConvAI at SemEval-2019 Task 6: Offensive language identification and categorization with perspective and BERT. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ted Pedersen. 2019. Duluth at SemEval-2019 Task 6: Lexical approaches to identify and categorize offensive tweets. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Andraž Pelicon, Matej Martinc, and Petra Kralj Novak. 2019. Embeddia at SemEval-2019 Task 6: Detecting hate with neural network and transfer learning approaches. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technology (NAACL-HLT)*.

Gretel Liz De la Pea and Paolo Rosso. 2019. DeepAnalyzer at SemEval-2019 Task 6: A deep learning-based ensemble method for identifying offensive tweets. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Andrei-Bogdan Puiu and Andrei-Octavian Brabete. 2019. Towards NLP with deep learning: Convolutional neural networks and recurrent neural networks for offensive language identification in social media. *arXiv preprint arXiv:1903.00665*.

Arun Rajendran, Chiyu Zhang, and Muhammad Abdul-Mageed. 2019. UBC-NLP at SemEval-2019 Task 6: Ensemble learning of offensive content with enhanced training data. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Murugesan Ramakrishnan, Wlodek Zadrozny, and Narges Tabari. 2019. UVA Wahoos at SemEval-2019 Task 6: Hate speech identification using ensemble machine learning. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Priya Rani and Atul Kr. Ojha. 2019. KMIColing at SemEval-2019 Task 6: Exploring n-grams for offensive language detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Alon Rozental and Dadi Biton. 2019. Amobee at SemEval-2019 Tasks 5 and 6: Multiple choice CNN over contextual embedding. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Jonathan Rusert and Padmini Srinivasan. 2019. NLP@UIOWA at SemEval-2019 Task 6: Classifying the classless using multi-windowed CNNs. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Angel Deborah S, Rajalakshmi S, Logesh B, Harshini S, Geetika B, Dyaneswaran S, S Milton Rajendram, and Mirnalinee T T. 2019. TECHSSN at SemEval-2019 Task 6: Identifying and categorizing offensive language in tweets using deep neural networks. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Silvia Sapora, Bogdan Lazarescu, and Christo Lolov. 2019. Absit invidia verbo: Comparing deep learning methods for offensive language. *arXiv preprint arXiv:1903.05929*.

Alessandro Seganti, Helena Sobol, Iryna Orlova, Hannam Kim, Jakub Staniszewski, Tymoteusz Krumholc, and Krystian Koziel. 2019. NLPR@SRPOL at SemEval-2019 Task 6 and Task 5: Linguistically enhanced deep learning offensive sentence classifier. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Elena Shushkevich, John Cardiff, and Paolo Rosso. 2019. TUVD team at SemEval-2019 Task 6: Offense target identification. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Pardeep Singh and Satish Chand. 2019. Pardeep at SemEval-2019 Task 6: Identifying and categorizing offensive language in social media using deep learning. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Murali Sridharan and Swapna T. 2019. Amrita School of Engineering - CSE at SemEval-2019 Task 6: Manipulating attention with temporal convolutional neural network for offense identification and classification. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Steve Durairaj Swamy, Anupam Jamatia, Björn Gambäck, and Amitava Das. 2019. NIT_Agartala_NLP_Team at SemEval-2019 Task 6: An ensemble approach to identifying and categorizing offensive language in Twitter social media corpora. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval)*.

Nithum Thain, Lucas Dixon, and Ellery Wulczyn. 2017. Wikipedia Talk Labels: Toxicity.

D Thenmozhi, Senthil Kumar B, Srinethe Sharavanan, and Aravindan Chandrabose. 2019. SSN_NLP at SemEval-2019 Task 6: Offensive language identification in social media using machine learning and deep learning approaches. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Johnny Torres and Carmen Vaca. 2019. JTML at SemEval-2019 Task 6: Offensive tweets identification using convolutional neural networks. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Harrison Uglow, Martin Zlocha, and Szymon Zmyslony. 2019. An exploration of state-of-the-art methods for offensive language detection. *arXiv preprint arXiv:1903.07445*.

Bin Wang, Xiaobing Zhou, and Xuejie Zhang. 2019. YNUWB at SemEval-2019 Task 6: K-max pooling cnn with average meta-embedding for identifying offensive language. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding abuse: A typology of abusive language detection subtasks. *arXiv preprint arXiv:1705.09899*.

Gregor Wiedemann, Eugen Ruppert, and Chris Biemann. 2019. UHH-LT at SemEval-2019 Task 6: Supervised vs. unsupervised transfer learning for offensive language detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 shared task on the identification of offensive language. In *Proceedings of the GermEval 2018 Workshop (GermEval)*.

Zhenghao Wu, Hao Zheng, Jianming Wang, Weifeng Su, and Jefferson Fong. 2019. BNU-HKBU UIC NLP Team 2 at SemEval-2019 Task 6: Detecting offensive language using BERT model. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from bullying traces in social media. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technology (NAACL-HLT)*.

Zhang Yaojie, Xu Bing, and Zhao Tiejun. 2019. CN-HIT-MI.T at SemEval-2019 Task6: Offensive language identification based on BiLSTM with double attention. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Predicting the type and target of offensive posts in social media. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technology (NAACL-HLT)*.

Chengjin Zhou, Jin Wang, and Xuejie Zhang. 2019. YNU-HPCC at SemEval-2019 Task 6: Identifying and categorising offensive language on Twitter. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Jian Zhu, Zuoyu Tian, and Sandra Kübler. 2019. UM-IU@LING at SemEval-2019 Task 6: Identifying offensive tweets using BERT and SVMs. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

NULI at SemEval-2019 Task 6: Transfer Learning for Offensive Language Detection using Bidirectional Transformers

Ping Liu
Department of Computer Science
Illinois Institute of Technology
pliu19@hawk.iit.edu

Wen Li
Department of Linguistics
Indiana University
wl9@indiana.edu

Liang Zou
Department of Mathematics
New York University
lz1904@nyu.edu

Abstract

Transfer learning and domain adaptive learning have been applied to various fields including computer vision (e.g., image recognition) and natural language processing (e.g., text classification). One of the benefits of transfer learning is to learn effectively and efficiently from limited labeled data with a pre-trained model. In the shared task of identifying and categorizing offensive language in social media, we preprocess the dataset according to the language behaviors on social media, and then adapt and fine-tune the Bidirectional Encoder Representation from Transformer (BERT) pre-trained by Google AI Language team[1]. Our team NULI wins **the first place (1st)** in Sub-task A - Offensive Language Identification and is ranked 4th and 18th in Sub-task B - Automatic Categorization of Offense Types and Sub-task C - Offense Target Identification respectively.

1 Introduction

Anti-social online behaviors, including cyberbullying, trolling and offensive language (Xu et al., 2012; Kwok and Wang, 2013; Cheng et al., 2017), are attracting more attention on different social networks. The intervention of such behaviors should be taken at the earliest opportunity. Automatic offensive language detection using machine learning algorithms becomes one solution to identifying such hostility and has shown promising performance.

In SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (Zampieri et al., 2019b), the organizers collected tweets through Twitter API and annotated them hierarchically regarding offensive language, offense type, and offense target. The task is divided into three sub-tasks: a) detecting if a post is offensive (OFF) or not (NOT); b) identifying the offense type of an offensive post as targeted insult (TIN), targeted threat (TTH), or untargeted (UNT); c) for a post labeled as TIN/TTH in sub-task B, identifying the target of offense as individual (IND), group of people (GRP), organization or entity (ORG), or other (OTH). The three sub-tasks are independently evaluated by macro-F1 metric.

[1]https://github.com/google-research/bert

The challenges of this shared task include: a) comparatively small dataset makes it hard to train complex models; b) the characteristics of language on social media pose difficulties such as out-of-vocabulary words and ungrammatical sentences; c) the distribution of target classes is imbalanced and inconsistent between training and test data. To address the problem of out-of-vocabulary words especially emoji and hashtags, we preprocess each tweet by interpreting emoji as meaningful English phrases and segmenting hashtags into space separated words. The classifiers we experiment with include: linear model with features of word unigrams, word2vec, and Hatebase; word-based Long Short-Term Memory (LSTM); fine-tuned Bidirectional Encoder Representation from Transformer (BERT) (Devlin et al., 2018). We choose BERT for our official submission, since it performs the best in our experiments.

In the rest of this paper, we organize the content as follows: related work of hostility on social media is stated in section 2; section 3 introduces data description, details of preprocessing, and the methodology of our models; experimental results are discussed in section 4. We also present the conclusion of our work at the end of paper.

2 Related Work

Schmidt and Wiegand (2017) surveyed features widely used for hate speech detection, including simple surface feature, word generalization, knowledge-based features, etc. Davidson et al. (2017) reported hate speech detection results using word n-grams and sentiment lexicon and provided insights on misclassified examples. A proposal of typology of abusive language sub-tasks is presented in (Waseem et al., 2017). (Liu et al., 2018) also discuss that the forecasting of the future hostility on Instagram can be divided into two levels: presence and intensity. In addition to English, researchers also investigated offensive language detection for Chinese (Su et al., 2017) and Slovene (Fišer et al., 2017). In the shared task on aggression identification organised as part of the first workshop on trolling, aggression and cyberbullying (TRAC - 1) at COLING 2018, word/character n-grams and word embeddings were the most commonly used features among the participants, and the most popular classifiers were SVM, LSTM, and RNN. The best performing system employed bidirectional LSTM on Glove embeddings.

3 Data and Methodology

3.1 Data Description

Offensive Language Identification Dataset (OLID) (Zampieri et al., 2019a) is collected from Twitter API by searching certain keywords set. The keywords include some unbiased targeted phrase such as 'she is', 'he is' and 'you are' which have high proportional offensive tweets. The distribution of offensive tweets is controlled around 30% by using different sampling methods. Another observation reported in the paper is political tweets tend to be more likely offensive using keywords as 'MEGA', 'liberal' and 'conservative'.

The main task of this competition is decomposed into three different levels according to the hierarchical annotation: a) Offensive Language Detection b) Categorization of Offensive Language c) Offensive Language Target Identification. All the three different tasks share the same dataset, and the latter one is the subset of the previous one.

The tasks release the dataset into three different parts, which are the startingKit, training dataset and testing dataset. The summary of dataset distribution is concluded in the Table1. From the table, it is easy to observe that the distribution of three splittings is a little twisted which should be expected in real life, and also make the tasks much harder.

Class	StartKit	Training	Testing
NOT	243	8840	620
OFF	77	4400	280
TIN	38	3876	213
UNT	39	524	27
IND	30	2407	100
GRP	7	1074	78
OTH	2	395	35

Table 1: Data Distribution: The first two rows are the class distribution of sub-task A. The mid part two rows are the class distribution of sub-task B. The last three rows are the class distribution of sub-task C.

3.2 Preprocessing

Emoji substitution We use one online emoji project on github [2] which could map the emoji unicode to substituted phrase. We treat such phrases into regular English phrase thus it could maintain their semantic meanings, especially when the dataset size is limited.

HashTag segmentation The HashTag becomes a popular culture cross multi social networks, including Twitter, Instagram, Facebook etc. In order to detect whether the HashTag contains profanity words, we apply word segmentation using one open source on the github [3]. One typical example would be '#LunaticLeft' is segmented as 'Lunatic Left' which is obviously offensive in this case.

Misc. We also convert all the text into lower case. 'URL' is substituted by 'http', since 'URL' does not have embedding representation in some pre-trained embedding and models. Consecutive '@USER's are limited to three times to reduce the redundancy.

3.3 Methodology

Linear model We firstly select Logistic Regression as our baseline model to determine the lower bound performance that we should compare. First we cross-validate hyper-parameters of different vectorizers to build bag of words representation. Secondly, we adopt the pre-trained word2vec model from google [4], then aggregate the maximum and average value in each dimension.

[2]https://github.com/carpedm20/emoji
[3]https://github.com/grantjenks/python-wordsegment
[4]https://code.google.com/archive/p/word2vec/

(a) Sub-task A

System	MacroF	Acc
All NOT	0.4004	0.6677
All OFF	0.2494	0.3323
Linear	0.7102	0.7273
LSTM	0.7166	0.7659
BERT	**0.7826**	0.8485

(b) Sub-task B

System	MacroF	Acc
All TIN	0.4686	0.8818
All UNT	0.1057	0.1182
Linear	**0.6028**	0.8000
LSTM	0.5029	0.8795
BERT	0.3830	0.8682

(c) Sub-task C

System	MacroF	Acc
All GRP	0.1441	0.2758
All IND	0.2554	0.6211
All OTH	0.0623	0.1031
Linear	0.5607	0.7062
LSTM	0.5056	0.7036
BERT	**0.8435**	0.7294

Table 2: Results on Dev Data.

Thirdly, we use the dictionary Hatebase API[5] to aggregate the hate words in each category. We validate all the features combinations, then report the accuracy and F1 with the highest to determine the model parameters.

LSTM Long Short-Term Memory is introduced in 1991 (Hochreiter and Schmidhuber, 1997) which is an more powerful extension of recurrent neural network. The gates inside of LSTM could prevent gradient vanishing problem, to memorize the long time dependency. LSTM has been used in tons of natural language processing task, such as sentiment classification, neural translation, language generation etc. We would also like to use LSTM as our second powerful baseline model to compare and report the result. The specific setting is the following: the input is mapped from one-hot encoder into a shared embedding layers with dimension 140; the hidden units of LSTM is 64 and follower by a dropout layer with rate 0.5. The maximum sequence length is 140, thus the sentences would be either cut off or padded.

BERT Google research team releases Bidirectional Encoder Representation from Transformer (BERT) (Devlin et al., 2018) and achieve state of the art results on many NLP tasks. BERT uses identical multi-head transformer structure that is introduced in (Vaswani et al., 2017). The model is pre-trained on huge corpus from different sources. Since the dataset size in this SemEval-2019 Task 6 is not that big, we pass the dataset into the pre-trained BERT model, and report the loss and accuracy at each epoch. The observation from experiments shows that after 1st or 2nd epochs, the model converges fast and always get very lower loss on the validation set. In such case, in the sub-task B and sub-task C, we report the macro-F1 score after the model trains after 1st, 2nd and 3rd epoch.

[5]http://www.hatebase.org

4 Experiment Results

The evaluation metric of this task is Macro-F1, which is the unweighted-average F1 of all the classes. The imbalance distribution makes the macro-F1 hard to achieve, and usually the score is penalized by the minority class. Weighted-loss is one solution during the training time to balance the model not to lead to the majority class prediction.

In the table 2 and 3, we report the results of our dev-dataset and final test dataset. From the table 2, we list the performance of our three selected models for each sub-task. The data is stratified split into 9:1 as train and test. There is also one independent validation set to determine the model selection that is split from train set. One observation from the table shows the problem of imbalanced data, so that higher accuracy does not guarantee higher macro-F1 score. Thus the stop criterion is based on average loss of validation set we mentioned before. Based on the results of validation, we choose to use BERT as our selected model for the final submission.

In the table 3, it shows the results on official test dataset. It should be noticed that in the sub-task A, we also submit one result of a Bagging classifier with number 50, and Logistic Regression is the weak classifier. The features are the same with linear model we mentioned before. The result from BERT model sub-task A achieves the 1st place among all the participants. BERT-3 denotes we train BERT with only 3 epochs. Same notation with the latter two sub-tables. In the sub-task B and sub-task C, the results are not as good as sub-task A due to two reasons: 1) the class distribution is more skewed than that of sub-task A. 2) the number of training instance is much smaller than sub-task A. The worst performance is sub-task C,

(a) Sub-task A

System	MacroF	Acc
All NOT	0.4189	0.7209
All OFF	0.2182	0.2790
Bagg	0.7558	0.8105
Linear	0.7501	0.7953
BERT-3	**0.8286**	**0.8628**

(b) Sub-task B

System	MacroF	Acc
All TIN	0.4702	0.8875
All UNT	0.1011	0.1125
BERT-1	0.6932	0.8875
BERT-2	0.4702	0.8875
BERT-3	**0.7159**	**0.8958**

(c) Sub-task C

System	MacroF	Acc
All GRP	0.1787	0.3662
All IND	0.2130	0.4695
All OTH	0.0941	0.1643
BERT-1	0.5267	**0.7277**
BERT-2	**0.5598**	0.6948

Table 3: Results on Test Data.

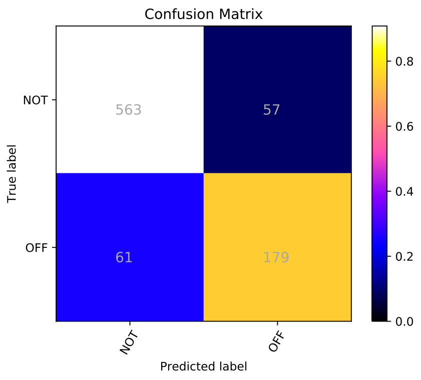


Figure 1: Sub-task A, BERT model after fine-tuning

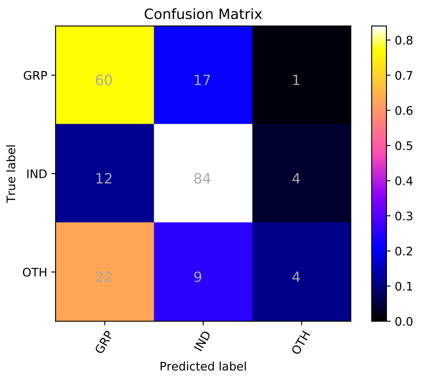


Figure 3: Sub-task C, BERT model after fine-tuning

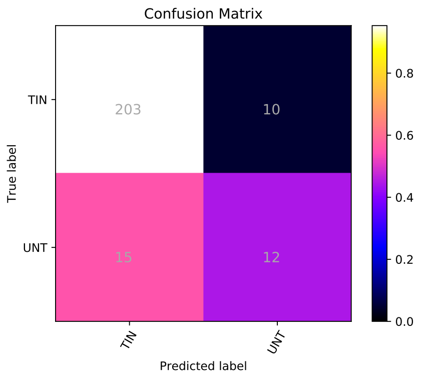


Figure 2: Sub-task B, BERT model after fine-tuning

since it is three-class classification, and the 'OTH' class has very few examples.

The confusion matrix of three sub-tasks are shown in fig 1, 2, and 3. This is another way to explain the results as we discussed before. The figures are provided by the organizers, and we use the figures to summarize test distribution in the table 1. In the previous section, we mentioned the discrepancy of class distribution between training and test datasets. For example, in sub-task C, the class 'OTH' constitutes 0.101 of the training data, while it makes up 0.164 of the test data. This adds difficulty to the task, however, we are often confronted with the same situation in real-world problems.

5 Conclusion

Offensive language and online hostility is crucial on the social network. The minority proportion of the nature and morphological language are the difficulties to achieve high performance. The Diversity and evolution of the language at different ages is another challenge for social media detection task. As a conclusion, our work shows the competitive results in this shared task using customized processing to dataset, as well as the power of pre-trained model. In real life, labeled data is always limited and requires expensive human labors. In such case, transfer learning is always a good option to get started. Domain adaption also has prior knowledge of specific domain before doing any modeling work on hand. How to tune the parameters is nontrivial, and there are a lot of more efficient ways to be explored, which could yield better performance.

References

Justin Cheng, Michael Bernstein, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. 2017. Anyone can become a troll: Causes of trolling behavior in online discussions. In *Proceedings of the 2017 ACM conference on computer supported cooperative work and social computing*, pages 1217–1230. ACM.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Darja Fišer, Tomaž Erjavec, and Nikola Ljubešić. 2017. Legal Framework, Dataset and Annotation Schema for Socially Unacceptable On-line Discourse Practices in Slovene. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Irene Kwok and Yuzhou Wang. 2013. Locate the hate: Detecting Tweets Against Blacks. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Ping Liu, Joshua Guberman, Libby Hemphill, and Aron Culotta. 2018. Forecasting the presence and intensity of hostility on instagram using linguistic and social features. In *Twelfth International AAAI Conference on Web and Social Media*.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Huei-Po Su, Chen-Jie Huang, Hao-Tsung Chang, and Chuan-Jie Lin. 2017. Rephrasing Profanity in Chinese Text. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from bullying traces in social media. In *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 656–666. Association for Computational Linguistics.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

CUNY-PKU Parser at SemEval-2019 Task 1: Cross-lingual Semantic Parsing with UCCA

Weimin Lyu[†], **Sheng Huang**[‡], **Abdul Rafae Khan**[†], **Shengqiang Zhang**[‡] , **Weiwei Sun**[‡], **Jia Xu**[†◇]

Computer Science Department, [†]Graduate Center and ◇Hunter College, City University of New York
{wlyu,akhan4}@gradcenter.cuny.edu, Jia.Xu@hunter.cuny.edu

[‡]Institute of Computer Science and Technology, Peking University
{huangsheng,ws,sq.zhang}@pku.edu.cn

Abstract

This paper describes the systems of the CUNY-PKU team in "SemEval 2019 Task 1: Cross-lingual Semantic Parsing with UCCA"[1]. We introduce a novel model by applying a cascaded MLP and BiLSTM model. Then, we ensemble multiple system-outputs by reparsing. In particular, we introduce a new decoding algorithm for building the UCCA representation. Our system won the first place in one track (French-20K-Open), second places in four tracks (English-Wiki-Open, English-20K-Open, German-20K-Open, and German-20K-Closed), and third place in one track (English-20K-Closed), among all seven tracks.

1 Introduction

We participate in all seven tracks in Cross-lingual Semantic Parsing at SemEval 2019. Our submission systems[2] are based on BiLSTM using TUPA (Hershcovich et al., 2017a, 2018).

Then, we built a second single parser using BiLSTM (Bi-directional LSTM) and Multi-Layer Perceptron (MLP) with TUPA (Hershcovich et al., 2017a, 2018). Most importantly, we introduce a new model Cascaded BiLSTM by first pre-training the BiLSTM and MLP model and then to continue training another MLP model. The cascaded BiLSTM parser significantly enhances the parsing accuracy on all tasks. We also complete a Self-Attentive Constituency Parser (Kitaev and Klein, 2018a,b) as comparison. Finally, we ensemble different parsers with a reparsing strategy (Sagae and Lavie, 2006). In particular, we introduce a novel algorithm based on dynamic programming to perform inference for the UCCA representation. This *decoder* can also be utilized as a core engine for a single parser.

In the post-evaluation stage, our improved systems are ranked first in three tracks (French-20K-Open, English-20K-Open and English-Wiki-Open) and second in the other four tracks. A shared task summary paper (Hershcovich et al., 2019) by competition organizers summaries the results.

We will describe our systems in detail, including three single parsers in Section 2 and a voter in Section 3. We focus on two novel technical contributions: the Cascaded BiLSTM model and the Reparsing strategy. In Section 4 we will present experimental setup and results.

2 Single Parsers

2.1 TUPA Parsers

The TUPA parser (Hershcovich et al., 2017a) builds on discontinuous constituency and dependency graph parsing and makes some improvements especially for the UCCA representation. The English parsing is based on Hershcovich et al. (2017a), while French and German parsing is based on Hershcovich et al. (2018).

It has been shown that the choice of model plays an important role in transition-based parsing (Hershcovich et al., 2017b). For TUPA, we built parsers with different models: MLP, BiLSTM, and also invent a new architecture, viz. Cascaded BiLSTM. The three single parsers are described as the following:

The MLP parser (Hershcovich et al., 2017b) applies a feedforward neural network with dense embedding features to predict optimal transitions given particular parser states. This parser adopts a similar architecture to Chen and Manning (2014).

The BiLSTM parser (Hershcovich et al., 2018) applies a bidirectional LSTM to learn con-

[1]https://competitions.codalab.org/competitions/19160

[2]https://github.com/weimin17/semEval-taks1

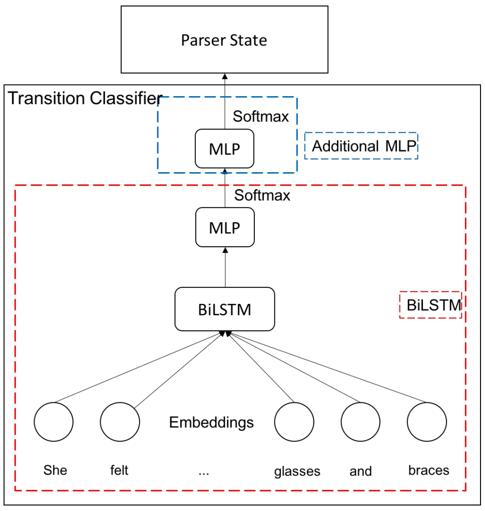


Figure 1: Illustration of the multi-stage Cascaded BiLSTM model. Top: parser state. Bottom: BiLTSM with two MLP architectures. The red box represents BiLSTM (Hershcovich et al., 2018), and the blue box represents a MLP that we add after implementing the BiLSTM architecture. Vector representation for the input tokens is computed by two layers of bidirectional LSTMs then fed into the double MLP with Softmax to select the next transition.

textualized vector-based representations for words that are then utilized for encoding a parser state, similarly to Kiperwasser and Goldberg (2016). The red box in Figure 1 shows the architecture of BiLSTM model, indicating that the representations after BiLSTM are fed into a Multiple-layer perceptron.

The Cascaded BiLSTM parser combines the above two parsing models, which contains a multi-stage training process. First, we use BiLSTM TUPA model to train 100 epochs, then retrain the model using MLP TUPA model for another 50 epochs. It's really interesting that the performances remains as good as BiLSTM TUPA model, even slightly better. Figure 1 shows the architecture of Cascaded BiLSTM model.

2.2 Phrase Constituency Parser

We also built a Constituency Parser as comparison, which uses a self-attentive architecture that makes explicit the manner considering information propagating between different locations in the sentences (Kitaev and Klein, 2018a,b). The constituency parser uses parsing tree structures as input and output. Therefore, we convert the phrase structure tree format into UCCA XML formation and vice versa.

3 The Reparsing System

The reparsing system (voter) takes multiple single parser (as in Section 2) results as input and produces a single, hopefully, improved UCCA graph as output. Briefly, each input UCCA graph is encoded to a chart of scores for standard CKY decoding. In this step, we utilize a number of auxiliary labels to encode remote edges and discontinuous constructions. These scores are summed up to get a new chart, which is used for CKY decoding for an immediate tree representation as the *voting* result. An immediate tree is then enhanced with reference relationships. Finally, a UCCA graph is built via interpreting auxiliary labels.

Span representation Graph nodes in a UCCA graph naturally create a hierarchical structure through the use of *primary edges*. Following this tree structure, we give the definition of span of nodes.

Definition 1. The span of node x is:

1. empty if x is an implicit node;
2. $[p, p+1)$ if x is a leaf node but not an implicit node, where p is the position of the lexical unit corresponding to x;
3. the union of spans of x's children, otherwise.

Assuming that each span of nodes is consecutive (we will deal with nonconsecutive spans in Section 3). We encode the label of edge from x's parent to x as the label of span of x. If there are some implicit nodes in x's children, the labels of edges from x to them are also encoded by the label of the span of x. If the span of x is the same as x's parent, the label of this span will be encoded ordered. This process is well-defined due to the acyclic graph structure. Each parser is assigned a weight to indicate its *contribution* to reparsing. The spans with labels encoded from a UCCA graph are assigned the same score according to which parser they come from. Thus, there is a set of scored spans for each UCCA graph. Following the parsing literature, we call this set a chart. We merge multiple charts produced by different parsers to a single chart simply by adding the corresponding scores.

Handling Remote Edges A remote edge with label L from node x to node y is equal to a primary edge with label L from x to an implicit node, which is referred to node y. Hence, if we can find the relationships of references, the remote edges are able to be recovered.

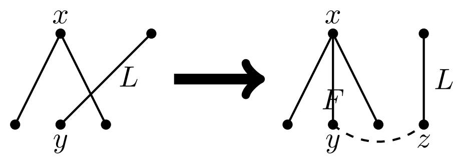


Figure 2: Remove nonconsecutive spans

Since all primary edges from nodes to their parent are encoded in labels of spans, each node could be represented as part of the label of a span. We encode each reference of a remote edge as a pair of two nodes with a score. After building all primary edges through dynamic programming, we search for available references with the maximum score in each implicit node greedily and leverage these references to recover remote edges.

Handling Discontinuous Spans Discontinuous spans are removed by repeating the following steps:

Step 1. Find a node x with a nonconsecutive span with the minimum starting point and minimum height, supposed its consecutive sub-span with minimum starting point is $[a, b)$.

Step 2. Find a node y with a consecutive span with starting point b and maximum height, supposed the primary edge from y's parent to y is e.

Step 3. Create a node z with a special type MIRROR and create a primary edge with the label of e from y's parent to z. Remove the primary edge e and create a primary edge with a special label FAKE from x to y.

After each iteration, the span of y is added to x, and the sum of the length of nonconsecutive spans decreases. Each primary edge in an original UCCA graph can only be removed once. To that end, the running time of this algorithm is linear in the number of lexical units. If all references of MIRROR nodes are correctly predicted, the expected UCCA graph will be obtained. In this way, remote edges can be handled.

4 Experiments

4.1 Data Statistics

The semantic parsing task is carried out in three languages: English, German and French, including three training data sets and parallel four test data sets. For English data, we use the Wikipedia UCCA corpus (henceforth Wiki) as training and development data, testing on English UCCA Wikipedia corpus as the in-domain test. Meanwhile, English UCCA Twenty Thousand Leagues Under the Sea English-French-German parallel corpus (henceforth 20K Leagues) serves as an out-of-domain test set. For German data, we use 20K Leagues corpus for train, development, and test sets. For French data, they provide only limited training data, along with development and test data sets.

Tracks	Training		Dev	Test
	Closed	Open		
En-Wiki	4113	5132	514	515
En-20K	0	5132	0	492
Ge-20K	5211	6360	651	632
Fr-20K	15	547	238	239

Table 1: Sentence number in training, dev, and test sets for English, German and French UCCA data sets.

Table 1 shows the sentences number of data sets for all three languages. We use the closed track data and UCCA's annotation resources for open tracks. We merge those resources and build our open track data[3].

4.2 TUPA Parsers

We build MLP and BiLSTM systems using TUPA (Hershcovich et al., 2017b). For Cascaded BiLSTM model, we add another MLP after the BiLSTM model, which forms a cascaded BiSLTM. For closed tracks, we train models based on the gold-standard UCCA annotation from official resources. For open tracks, We use additional UCCA data from other open sources as training data set. We also generate synthetic data by automatically translating text (Khan et al., 2018) and its parsing labels across languages in our on-going work.

Table 2 shows the results for four models in different tracks. The italicized values are our official submission. However, we have made some improvement after the Evaluation Phrase, and the bold results are our best results. The first three models are single systems and the fourth model (Ensembled) ensembles different frameworks by reparsing systems. The "baseline" represents the baseline that competition provides for reference.

By using feedforward Neural Network and embedding features, MLP models get the lowest scores. BiLSTM models achieve better results than MLP models in F1 scores, both in the in-domain and out-of-domain data sets. However, the

[3]`https://github.com/weimin17/semEval-taks1`

Tracks		MLP	BiLSTM(Submit)	Cascaded BiLSTM	Ensembled	baseline
closed	En-Wiki	0.650	*0.718*	0.721	**0.728**	0.728
	En-20K	0.617	*0.669*	0.673	**0.681**	0.672
	Ge-20K	0.699	*0.797*	**0.797**	0.797	0.731
open	En-Wiki	0.784	*0.800*	0.843	**0.846**	0.735
	En-20K	0.715	*0.739*	0.764	**0.770**	0.684
	Ge-20K	0.598	*0.841*	**0.841**	0.840	0.791
	Fr-20K	0.535	*0.796*	0.795	**0.796**	0.487

Table 2: F1 scores for both closed and open tracks in SemEval Task 1 2019 competition. The italic text represents our official submission in competition and the bold text represents our best F1 scores.

Open Tracks	F1 Scores
English-Wiki	0.75
English-20K	0.785

Table 3: F1 scores on unlabeled data.

Models	test sets	dev sets
CharsLSTM	91.96	92.21
ELMO	94.31	94.75

Table 4: F1 scores for two constituency parsers on both Penn Treebank dev and test data sets.

combination of BiSLTM and MLP models (Cascaded BiLSTM model) performs best among the three models in all results of single systems.

Our in-house reparsing system ensembles the above parsers as described in Section 3. We can see that ensemble results are better at closed track, but not as good as the best results by Cascaded BiLSTM at Open track.

4.3 Phrase Constituency Parser

For Phrase Constituency Parer, we only test the performance on unlabeled data instead of labeled data, while for TUPA we test on labeled data.

First, we use Benepar[4], a parsing tool using out-of domain pre-trained models to predict the labels, the outputs are parsing tree structures.

Second, we convert the constituency parsing tree structure to Conllu Format. We develop a one-shot tool to improve the efficiency of conversion based on TreebankPreprocessing[5], which can automatically convert a batch of files in one directory.

Finally, we convert Conllu format to UCCA XML using format[6].

We only experiment on English-Wiki and English-20K open track, and the results are pretty bad, as shown in Table 3. We hypothesize there are two reasons: 1. The conversion process could unavoidably cause accuracy loss. 2. The third-party pre-trained models are not as efficient as the models trained directly on the specific UCCA data.

To test the accuracy on unlabeled data and to evaluate how many losses are there during the conversion process, we evaluate the accuracy in the parsing tree structure phrase before the conversion. We experimentally validate our system on the English Wiki data set. We use official training data set as training data, splitting official dev set into two parts and separately serving as our dev set and test set. We also use two models of constituency parser: ELMO and CharsLSTM, tested on Penn Treebank (Cross and Huang, 2019) data.

Table 4 indicates that, for Penn Treebank data sets, CharsLSTM model's F1 score achieves 92.21 on dev data set, with F1 score 91.96 on the test dataset. Using ELMo, The dev dataset's F1 score achieves 94.75, with F1 score achieves 94.31 on test data set.

5 Summary

Our submission systems mainly contain a BiLSTM, an MLP, and a cascaded BiLSTM parser, as well as a voted system of above. Our final system ranks first in three tracks, French-20K-Open, English-20K-Open and English-Wiki-Open, and the second place in the other four tracks in the post-evaluation.

[4]https://github.com/nikitakit/self-attentive-parser

[5]https://github.com/hankcs/TreebankPreprocessing

[6]https://github.com/huji-nlp/semstr/blob/master/semstr/convert.py

Contributions and Acknowledgements

Weimin Lyu: built all TUPA Parsers, a self-attentive Parser, convert UCCA graph as constituency tree, verify the voting systems, and draft the paper. **Sheng Huang** and **Shengqiang Zhang**: built the reparsing system and UCCA-Dependency graph transformer. **Abdul Rafae Khan**: built cross-lingual parsers by generating synthetic data with machine translation. **Weiwei Sun**: extensively supervised PKU team, and **Jia Xu**: closely supervised CUNY team, in algorithms and experiments. We thank the initial work of Mark Perelman. This research was partially funded by National Science Foundation (NSF) Award No. 1747728 and National Science Foundation of China (NSFC) Award No. 61772036 and 61331011 and partially supported by the Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology) and the Computer Science Department at CUNY Graduate Center as well as CUNY Hunter College.

References

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

James Cross and Liang Huang. 2019. Span-based constituency parser.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017a. A transition-based directed acyclic graph parser for ucca. In *Proc. of ACL*, pages 1127–1138.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017b. A transition-based directed acyclic graph parser for ucca. *arXiv preprint arXiv:1704.00552*.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018. Multitask parsing across semantic representations. *arXiv preprint arXiv:1805.00287*.

Daniel Hershcovich, Zohar Aizenbud, Leshem Choshen, Elior Sulem, Ari Rappoport, and Omri Abend. 2019. Semeval 2019 task 1: Cross-lingual semantic parsing with ucca. *arXiv preprint arXiv:1903.02953*.

Abdul Khan, Subhadarshi Panda, Jia Xu, and Lampros Flokas. 2018. Hunter nmt system for wmt18 biomedical translation task: Transfer learning in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 655–661.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Nikita Kitaev and Dan Klein. 2018a. Constituency parsing with a self-attentive encoder. *arXiv preprint arXiv:1805.01052*.

Nikita Kitaev and Dan Klein. 2018b. Multilingual constituency parsing with self-attention and pre-training. *arXiv preprint arXiv:1812.11760*.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*.

DANGNT@UIT.VNU-HCM at SemEval 2019 Task 1: Graph Transformation System from Stanford Basic Dependencies to Universal Conceptual Cognitive Annotation (UCCA)

Dang Tuan Nguyen and Trung Tran
University of Information Technology, VNU-HCM
Ho Chi Minh City, Vietnam
dangnt@uit.edu.vn, ttrung@nlke-group.net

Abstract

This paper describes the graph transformation system (GT System) for SemEval 2019 Task 1: Cross-lingual Semantic Parsing with Universal Conceptual Cognitive Annotation (UCCA)[1]. The input of GT System is a pair of text and its unannotated xml, which is a layer 0 part of UCCA form. The output of GT System is the corresponding full UCCA xml. Based on the idea of graph illustration and transformation, we perform four main tasks when building GT System. At the first task, we illustrate the graph form of stanford dependencies[2] of input text. We then transform into an intermediate graph in the second task. At the third task, we continue to transform into ouput graph form. Finally, we create the output UCCA xml.

The evaluation results show that our method generates good-quality UCCA xml and has a meaningful contribution to the semantic representation sub-field in Natural Language Processing.

1 Introduction

In the past few years, semantic representation is receiving growing attention in NLP. Researchers have recently proposed different semantic schemes. Examples include Abstract Meaning Representation (Banarescu et al. 2013), Broad-coverage Semantic Dependencies (Oepen et al. 2014), Universal Decompositional Semantics (White et al. 2016), Parallel Meaning Bank (Abzianidze et al. 2016), Universal Conceptual Cognitive Annotation (Abend and Rappoport 2013). These advances in semantic representation, along with corresponding advances in semantic parsing, text understanding, summarization, paraphrase detection, and semantic evaluation.

In SemEval 2019 Task 1: Cross-lingual Semantic Parsing with Universal Conceptual Cognitive Annotation (UCCA)[1], the Committee focuses on parsing text according to the UCCA semantic annotation. UCCA (Abend and Rappoport 2013) is a cross-linguistically applicable semantic representation scheme, based on Basic Linguistic Theory (Dixon 2010). In general, UCCA represents the semantics of linguistic utterances as directed acyclic graphs (DAGs). In one DAG, nodes and edges belong to one of several layers. There are two types of node: (i) terminal nodes express the text tokens; (ii) non-terminal nodes express semantic units. Edges are labelled, indicating the role of a child in the relation the parent represents. As an example, consider sentence in Example 1: "*The album was recorded in Switzerland .*". Two layers of UCCA xml of this sentence:

- Layer0:

```
<root annotationID="0" passageID="503012">
    <attributes />
    <layer layerID="0">
      <attributes />
      <extra ... />
      <node ID="0.1" type="Word">
        <attributes ... text="The" />
        <extra dep="det" ... tag="DT" />
      </node>
      ...
    </layer>
</root>
```

The relations of NodeID and corresponding lexicon:

```
{[ID="0.1" ➔ dep="det" ➔ "The"]
[ID="0.2" ➔ dep="nsubj:pass" ➔ "album"]
[ID="0.3" ➔ dep="aux:pass" ➔ "was"]
[ID="0.4" ➔ dep="root" ➔ "recorded"]
[ID="0.5" ➔ dep="case" ➔ "in"]
[ID="0.6" ➔ dep="obl" ➔ "Switzerland"]
[ID="0.7" ➔ dep="punct" ➔ "."]}
```

- Layer1:

```
<layer layerID="1">
      <attributes />
      <node ID="1.1" type="FN">
        <attributes />
```

[1] https://competitions.codalab.org/competitions/19160

```
      <edge toID="1.2" type="H">
        <attributes />
      </edge>
    </node>
    <node ID="1.2" type="FN">
      <attributes />
      <edge toID="1.4" type="A">
        <attributes />
      </edge>
      ...
    </node>
    <node ID="1.4" type="FN">
      <attributes />
      <edge toID="1.10" type="E">
        <attributes />
      </edge>
      ...
    </node>
    <node ID="1.10" type="FN">
      <attributes />
      <edge toID="0.1" type="Terminal">
        <attributes />
      </edge>
    </node>
    ...
  </layer>
```

We have the graphical representation of the above UCCA:

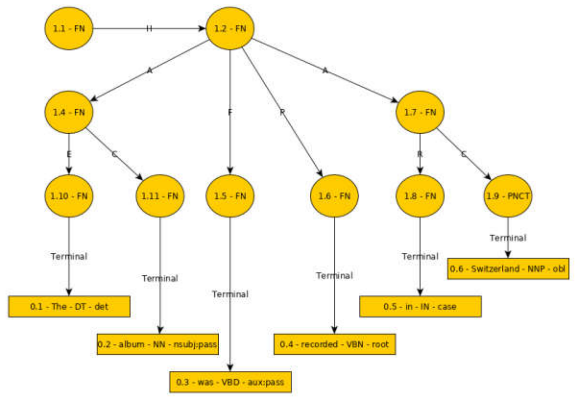


Figure 1: Graph form of UCCA xml of sentence in Example 1.

The primary purpose of this article is to present our system called graph transformation system (GT System) for Task[1]. We perform four tasks when building GT System. At the first task, we illustrate the graph form of Stanford dependencies[2] (Manning et al. 2014; Marie-Catherine et al. 2014) of input text. We then transform into an intermediate graph in the second task. At the third task, we continue to transform into ouput graph form. Finally, we create the output UCCA xml.

The rest of article is separated as follows. We briefly describe Stanford dependencies in Section 2. In Section 3, we introduce our GT system for Task[1]. Section 4 details the experiments and analyzes the results. We offer conclusions in Section 5.

2 Stanford Dependencies

Stanford dependencies[2] (Manning et al. 2014; Marie-Catherine et al. 2014; Marie-Catherine and Manning 2008) provides a representation of grammatical relations between words in a sentence. Stanford dependencies (SD) have three parts: name of the relation, governor and dependent. Consider English sentence in Example 1, below is the xml representation of SD basic dependencies. This representation is the result of running Stanford CoreNLP pipeline (Manning et al. 2014).

```
<dependencies type="basic-dependencies">
     <dep type="root">
       <governor idx="0">ROOT</governor>
       <dependent idx="4">recorded</dependent>
     </dep>
     <dep type="det">
       <governor idx="2">album</governor>
       <dependent idx="1">The</dependent>
     </dep>
     <dep type="nsubjpass">
       <governor idx="4">recorded</governor>
       <dependent idx="2">album</dependent>
     </dep>
     <dep type="auxpass">
       <governor idx="4">recorded</governor>
       <dependent idx="3">was</dependent>
     </dep>
     <dep type="case">
       <governor
idx="6">Switzerland</governor>
       <dependent idx="5">in</dependent>
     </dep>
     <dep type="nmod">
       <governor idx="4">recorded</governor>
       <dependent
idx="6">Switzerland</dependent>
     </dep>
</dependencies>
```

We have the graphical representation of the above SD basic dependencies:

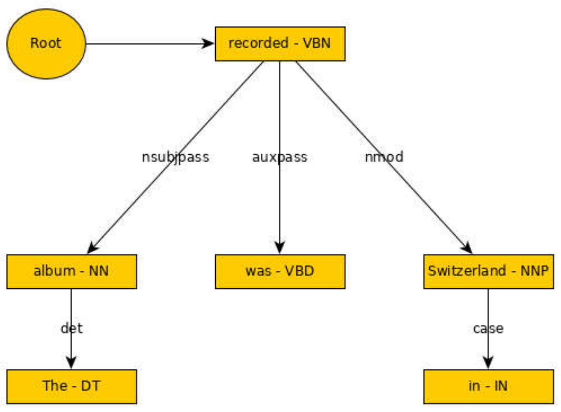


Figure 2: Graph form of SD basic dependencies of sentence in Example 1.

[2] https://stanfordnlp.github.io/CoreNLP/

3 The Graph Transformation System

In this section, we express our GT system for creating UCCA xml of the input text. The general architecture is represented in Figure 2:

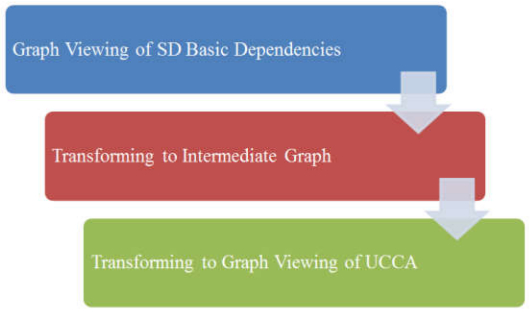

Graph Viewing of SD Basic Dependencies
Transforming to Intermediate Graph
Transforming to Graph Viewing of UCCA

Figure 3: Architecture of Graph Transformation System.

When building GT System, we perform two processes: training and testing process. At training process, we build the intermediate graph from UCCA and SD basic dependencies of training data[1]. At the testing process, which can be called the inverse process of training, we build the ouput UCCA from intermediate graph of testing data.

3.1 Intermediate Graph

In general, the intermediate graph is an irreducible representation of UCCA graph form. This intermediate graph is quite similar to graph form of SD basic dependencies. The main difference of the intermediate graph and graph form of SD basic dependencies is: each edge label in the intermediate graph is the combination of UCCA categories (Abend and Rappoport. 2013) and Stanford dependency relations (Marie-Catherine and Manning 2008a, 2008b).

Below is the intermediate graph of sentence in Example 1. This graph is the reduction of graph in Figure 1, and quite similar to graph in Figure 2.

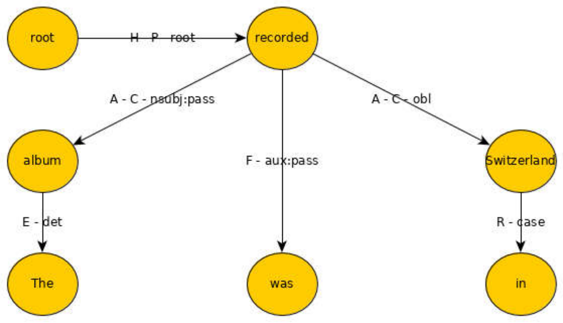


Figure 4: Intermediate graph of sentence in Example 1.

3.2 Training Process

Firstly, at training process, we consider train data[1] and performed main tasks. The first and second task is in turn viewing the graph from of SD basic dependencies and UCCA of input text. At the third task, we propose Left-First-Search liked algorithm with Bottom-Up idea to reduce the graph form of UCCA to intermediate graph. At the final task, we propose rules and heuristics for matching graph form of SD basic dependencies and intermediate graph.

The main steps of Left-First-Search (LFS) algorithm is as follow. ***Step 1***. Browse to terminal on the left. ***Step 2***. Back to parent node of this terminal. Check if parent having any other child or not. Step 2.1. If yes. Repeat Step 1 with root is this child node. ***Step 3***. Swap the position of root of sub-tree with position of child having important annotation. ***Step 4***. Back to parent node of this root. Repeat Step 2 with this parent.

To perform LFS algorithm, we determine the priority of SD and UCCA annotations according to two factors. **First**. The meaning of each annotation, representing the dependency relations and grammatical roles of lexicons. **Second**. The position of each node in graph.

Apply LFS algorithm for graph in Figure 3, we in turn have three level reductions in Figure 5, 6, 4 (respectively):

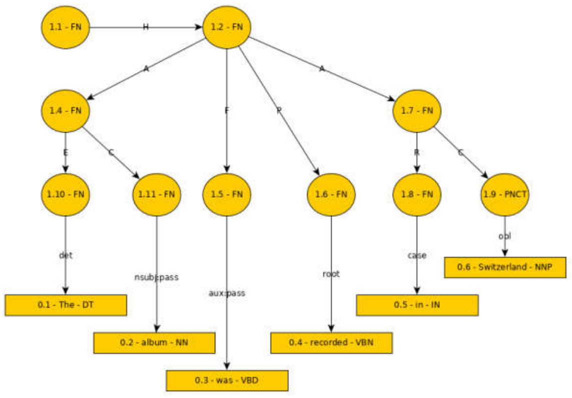


Figure 5: First reduction of Graph form in Figure 1.

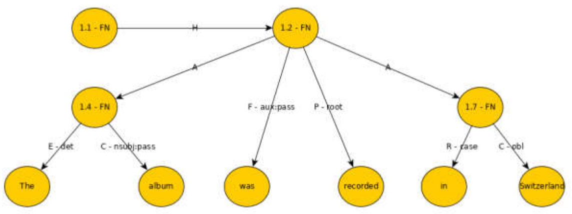


Figure 6: Second reduction of Graph form in Figure 1.

After having the final reduction, which is intermediate graph, of graph form of UCCA, we compare with graph form of SD basic dependencies. We consider the similarities between two graphs and propose rules and heuristics to (i) determine the level of one node, and (ii) determine the group of UCCA annotation for each level. The general idea of mechanism is:

- Collect all SD-type of relations in UCCA and SD basic dependencies of training data. Below is the collection:

SD basic dependencies	acl:relcl / expl / csubjpass / cop / aux / conj / acl / xcomp / dep / appos / advmod / neg / det / cc:preconj / nmod:tmod / ccomp / root / advcl / nsubj / case / iobj / cc / det:predet / nmod:poss / compound:prt / csubj / nsubjpass / nummod / nmod:npmod / nmod / auxpass / parataxis / amod / compound / discourse / mwe / dobj / mark
UCCA	acl:relcl / expl / obl:npmod / cop / aux / conj / acl / appos / xcomp / goeswith / advmod / det / ccomp / nsubj:pass / cc:preconj / nmod:tmod / flat / root / obl:tmod / advcl / punct / nsubj / case / iobj / cc / vocative / det:predet / nmod:poss / compound:prt / csubj / nummod / nmod:npmod / nmod / parataxis / amod / list / compound / discourse / aux:pass / obj / obl / fixed / mark

- Determine the priority order of SD-type relations.

Example 2: dobj -> amod -> dep -> nmod -> case.

- Determine the compound (UCCA and SD) relation in each node level.

Example 3: type conj at level 7: "H - A - E - C - C - C - conj"

3.3 Testing Process

At the testing process, which can be called the inverse process of training, we considered development and test data[1] and performed main tasks. The first task is viewing the graph from of SD basic dependencies of input text. At the second task, we applied proposed rules and heuristics to transform this graph to intermediate graph. We then, at the final task, we proposed Breadth-First-Search liked algorithm with Top-Down idea to re-create the graph form of UCCA from intermediate graph. This BFS algorithm is, in fact, the inverse mechanism of LFS algorithm in Section 3.2.

The main steps of Breadth-First-Search (BFS) algorithm is as follow. ***Step 1***. Reduce the first level of node. ***Step 2***. Determine the intergrated-Child which adheres to this node. ***Step 3***. If there is no intergratedChild. Step 3.1. Repeat Step 1 until node come down to terminal position. Step 3.2. Repeat from Step 1 to Step 4 with each child of this node. ***Step 4***. If there is intergratedChild. Step 4.1. Repeat from Step 1 to Step 4 with each child of this node which are different from intergrated-Child. Step 4.2. Repeat from Step 1 to Step 4 with this node. Step 4.3. Repeat from Step 1 to Step 4 with intergratedChild.

4 Experiment and Evaluation

At the evaluation phase, we focus on English in-domain setting, using the Wiki corpus. In testing data, this domain consists of 515 small texts with corresponding unannotated UCCA xmls.

We test our method for both open and closed track in the English setting: (i) closed track submission is only allowed to use the gold-standard UCCA annotation distributed for the task in the target language, and limited in its use of additional resources; (ii) open track submission is allowed to use any additional resource.

Table 1 and 2 view the results of testing data for open and closed tracks with labeled (first row) and unlabeled scores (second row).

Averaged F1		**P**	**R**	**F1**
0.708	Primary	0.738	0.694	0.715
	Remote	1.000	0.000	0.000
0.822	Primary	0.857	0.806	0.831
	Remote	1.000	0.000	0.000

Table 1: Results of testing data in open track.

Averaged F1		**P**	**R**	**F1**
0.706	Primary	0.737	0.692	0.714
	Remote	1.000	0.000	0.000
0.825	Primary	0.860	0.808	0.833
	Remote	1.000	0.000	0.000

Table 2: Results of testing data in closed track.

The testing results show that our GT system creates good quality UCCA semantic representations in English Wiki testing data.

5 Conclusion

We have presented the graph transformation method for creating UCCA semantic representation from English in-domain setting, using the Wiki corpus[1]. Our method performs four main tasks: (i) illustrate the graph form of Stanford dependencies[2] of input text; (ii) transform into an intermediate graph; (iii) continue to transform into ouput graph form; (iv) create the output UCCA xml. The experiment results show that our method meets the requirements from SemEval Task[1].

In future works, we intend to improve the transformational algorithms and propose more accurate rules for selecting best nodes and dependency tags. Besides, we expand our method and test with other datasets for a broader comparison.

References

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, Benjamin Van Durme. 2016. Universal Decompositional Semantics on Universal Dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas, pages 1713–1723.

Daniel Hershcovich, Omri Abend and Ari Rappoport. 2017. A Transition-Based Directed Acyclic Graph Parser for UCCA. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Vancouver, Canada, pages 1127–1138.

Daniel Hershcovich, Omri Abend and Ari Rappoport. 2018. Multitask Parsing Across Semantic Representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, pages 373–385.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *LREC 2016*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. Association for Computational Linguistics, pages 178–186.

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, Johan Bos. 2017. The Parallel Meaning Bank: Towards a Multilingual Corpus of Translations Annotated with Compositional Meaning Representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. Valencia, Spain, pages 242–247.

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55-60.

Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal Stanford Dependencies: A cross-linguistic typology. In *Proceedings of LREC*.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008a. The Stanford typed dependencies representation. In *COLING 2008 Workshop on Cross-framework and Cross-domain Parser Evaluation*.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008b. *Stanford Dependencies manual*.

Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sofia, Bulgaria, pages 228–238.

Omri Abend, Shai Yerushlami and Ari Rappoport. 2017. UCCAApp: Web-application for Syntactic and Semantic Phrase-based Annotation. In *Proceedings of ACL 2017*.

Robert M. W. Dixon. 2010. *Basic Linguistic Theory: Grammatical Topics, Volume 2*. Oxford University Press.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English Universal Dependencies: An Improved Representation for Natural Language Understanding Tasks. In *LREC 2016*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 Task 8:Broad-Coverage Semantic Dependency Parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland, pages 63–72.

GCN-Sem at SemEval-2019 Task 1: Semantic Parsing using Graph Convolutional and Recurrent Neural Networks

Shiva Taslimipoor **Omid Rohanian** **Sara Može**
Research Group in Computational Linguistics
University of Wolverhampton, UK
{shiva.taslimi, omid.rohanian, s.moze}@wlv.ac.uk

Abstract

This paper describes the system submitted to the SemEval 2019 shared task 1 'Cross-lingual Semantic Parsing with UCCA'. We rely on the semantic dependency parse trees provided in the shared task which are converted from the original UCCA files and model the task as tagging. The aim is to predict the graph structure of the output along with the types of relations among the nodes. Our proposed neural architecture is composed of Graph Convolution and BiLSTM components. The layers of the system share their weights while predicting dependency links and semantic labels. The system is applied to the CONLLU format of the input data and is best suited for semantic dependency parsing.

1 Introduction

Universal Conceptual Cognitive Annotation (UCCA) (Abend and Rappoport, 2013) is a semantically motivated approach to grammatical representation inspired by typological theories of grammar (Dixon, 2012) and Cognitive Linguistics literature (Croft and Cruse, 2004). In parsing, bi-lexical dependencies that are based on binary head-argument relations between lexical units are commonly employed in the representation of syntax (Nivre et al., 2007; Chen and Manning, 2014) and semantics (Hajič et al., 2012; Oepen et al., 2014; Dozat and Manning, 2018).

UCCA differs significantly from traditional dependency approaches in that it attempts to abstract away traditional syntactic structures and relations in favour of employing purely semantic distinctions to analyse sentence structure. The shared task, 'cross-lingual semantic parsing with UCCA' (Hershcovich et al., 2019) consists in parsing English, German, and French datasets using the UCCA semantic tagset. In order to enable multi-task learning, the UCCA-annotated data is automatically converted to other parsing formats, e.g. Abstract Meaning Representation (AMR) and Semantic Dependency Parsing (SDP), inter alia (Hershcovich et al., 2018).

Although the schemes are formally different, they have shared semantic content. In order to perform our experiments, we target the converted CONLLU format, which corresponds to traditional bi-lexical dependencies and rely on the conversion methodology which is provided in the shared task (Hershcovich et al., 2019) to attain UCCA graphs.

UCCA graphs contain both explicit and implicit units [1] However, in bi-lexical dependencies, nodes are text tokens and semantic relations are direct bi-lexical relations between the tokens. The conversion between the two format results in partial loss of information. Nonetheless, we believe that it is worth trying to model the task using one of the available formats (i.e. semantic dependency parsing) which is very popular among NLP researchers.

Typically, transition-based methods are used in syntactic (Chen and Manning, 2014) and semantic (Hershcovich et al., 2017) dependency parsing. By contrast, our proposed system shares several similarities with sequence-to-sequence neural architectures, as it does not specifically deal with parsing transitions. Our model uses word, POS and syntactic dependency tree representations as input and directly produces an edge-labeled graph representation for each sentence (i.e. edges and their labels as two separate outputs). This multi-label neural architecture, which consists of a BiLSTM and a Graph Convolutional Network (GCN), is described in Section 3.

[1] Explicit units (terminal nodes) correspond to tokens in the text, but implicit (semantic) units have no corresponding component in the text.

2 Related Work

A recent trend in parsing research is sequence-to-sequence learning (Vinyals et al., 2015b; Kitaev and Klein, 2018), which is inspired from Neural Machine Translation. These methods ignore explicit structural information in favour of relying on long-term memory, attention mechanism (content-based or position-based) (Kitaev and Klein, 2018) or pointer networks (Vinyals et al., 2015a). By doing so, high-order features are implicitly captured, which results in competitive parsing performance (Jia and Liang, 2016).

Sequence-to-sequence learning has been particularly effective in Semantic Role Labeling (SRL) (Zhou and Xu, 2015). By augmenting these models with syntactic information, researchers have been able to develop state-of-the-art systems for SRL (Marcheggiani and Titov, 2017; Strubell et al., 2018).

As information derived from dependency parse trees can significantly contribute towards understanding the semantics of a sentence, Graph Convolutional Network (GCN) (Kipf and Welling, 2017) is used to help our system perform semantic parsing while attending to structural syntactic information. The architecture is similar to the GCN component employed in Rohanian et al. (2019) for detecting gappy multiword expressions.

3 Methodology

For this task, we employ a neural architecture utilising structural features to predict semantic parsing tags for each sentence. The system maps a sentence from the source language to a probability distribution over the tags for all the words in the sentence. Our architecture consists of a GCN layer (Kipf and Welling, 2017), a bidirectional LSTM, and a final dense layer on top.

The inputs to our system are sequences of words, alongside their corresponding POS and named-entity tags.[2] Word tokens are represented by contextualised ELMo embeddings (Peters et al., 2018), and POS and named-entity tags are one-hot encoded. We also use sentence-level syntactic dependency parse information as input to the system. In the GCN layer, the convolution filters operate based on the structure of the dependency tree (rather than the sequential order of words).

Graph Convolution. Convolutional Neural Networks (CNNs), as originally conceived, are sequential in nature, acting as detectors of N-grams (Kim, 2014), and are often used as feature-generating front-ends in deep neural networks. Graph Convolutional Network (GCN) has been introduced as a way to integrate rich structural relations such as syntactic graphs into the convolution process.

In the context of a syntax tree, a GCN can be understood as a non-linear activation function f and a filter W with a bias term b:

$$c = f\left(\sum_{i \in r(v)} W x_i + b\right) \tag{1}$$

where $r(v)$ denotes all the words in relation with a given word v in a sentence, and c represents the output of the convolution. Using adjacency matrices, we define graph relations as mask filters for the inputs (Kipf and Welling, 2017; Schlichtkrull et al., 2017).

In the present task, information from each graph corresponds to a sentence-level dependency parse tree. Given the filter W_s and bias b_s, we can therefore define the sentence-level GCN as follows:

$$C_s = f(W_s X^T A + b_s) \tag{2}$$

where $X_{n\times v}$, $A_{n\times n}$, and $C_{o\times n}$ are tensor representation of words, the adjacency matrix, and the convolution output respectively.[3] In Kipf and Welling (2017), a separate adjacency matrix is constructed for each relation to avoid overparametrising the model; by contrast, our model is limited to the following three types of relations: 1) the head to the dependents, 2) the dependents to the head, and 3) each word to itself (self-loops) similar to Marcheggiani and Titov (2017). The final output is the maximum of the weights from the three individual adjacency matrices.
The model architecture is depicted in Figure 1.

4 Experiments

Our system participated in the closed track for English and German and the open track for French. We exclusively used the data provided in the shared task. The system is trained on the training data only, and the parameters are optimised using the development set. The results are reported

[2] spaCy (Honnibal and Johnson, 2015) is used to generate POS, named-entity and syntactic dependency tags.

[3] o: output dimension; v: word vectors dimension; n: sentence length

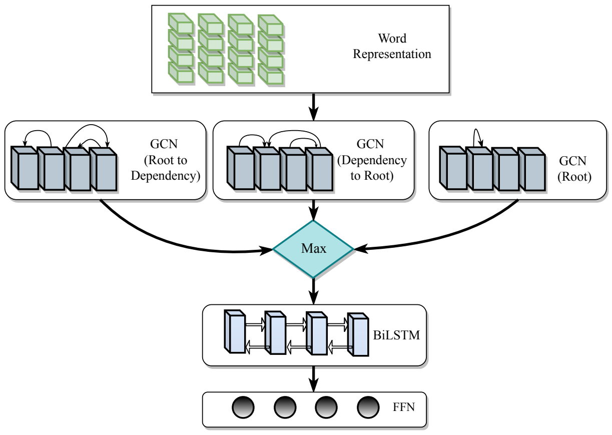


Figure 1: A GCN-based recurrent architecture.

on blind-test data in both in-domain and out-of-domain settings. We focus on predicting the primary edges of UCCA semantic relations and their labels.

4.1 Data

The datasets of the shared task are devised for four settings: 1) English in-domain, using the Wiki corpus; 2) English out-of-domain, using the Wiki corpus as training and development data, and 20K Leagues as test data; 3) German in-domain, using the 20K Leagues corpus; 4) French setting with no training data (except trial data), using the 20K Leagues corpus as development and test data.

Whilst the annotated files used by the shared task organisers are in the XML format, several other formats are also available. We decided to use CONLLU, as it is more interpretable. However, according to the shared task description,[4] the conversion between XML and CONLLU, which is a necessary step before evaluation, is lossy. Hershcovich et al. (2017) used the same procedure of performing dependency parsing methods on CONLLU files and converting the predictions back to UCCA.

4.2 Settings

We trained ELMo on each of the shared task datasets using the system implemented by Che et al. (2018). The embedding dimension is set to 1024. The number of nodes is 256 for GCN and 300 for BiLSTM, and we applied a dropout of 0.5 after each layer. We used the Adam optimiser for compiling the model.

We tested our model in four different settings, as explained in Section 4.1. The parameters are optimised on the English Wiki development data (batch-size = 16 and number of epochs = 100) and used for all four settings. As no training data was available for French, the trained system on English Wiki was used to parse French sentences of 20K Leagues. For this reason the French model is evaluated within the open track.

4.3 Official Evaluation

Our model predicts two outputs for each dataset: primary edges and their labels (UCCA semantic categories). [5]

Table 1 shows the performance (in terms of precision, recall, and F1-score) for predicting primary edges in both labeled (i.e. with semantic tags) and unlabeled settings (i.e. ignoring semantic tags). Table 2 shows F1-scores for each semantic category separately. Although the overall performance of the system, as shown in the official evaluation in Table 1, is not particularly impressive, there are a few results worth reporting. These are listed in Table 2.

Our system is ranked second in predicting four relations, i.e. L (linker), N (Connector), R (Relator), and G (Ground), in all settings displayed in bold. A plausible explanation would be that these relations are somewhat less affected by the loss of information incurred as a result of the conversions between formats.

5 Discussion

Our neural model is applied to UCCA corpora, which are converted to bi-lexical semantic dependency graphs and represented in the CONLLU format. The conversion from UCCA annotations to CONLLU tags appears to have a distinctly negative impact on the system's overall performance. As reported in the shared task description, converting the English Wiki corpus to the CONLLU format and back to the standard format results in an F1-score of only 89.7 for primary labeled edges. This means that our system cannot go beyond this upper limit.

Since our system is trained on CONLLU files and the evaluation involves converting the CONLLU format back to the standard UCCA format,

[4]https://competitions.codalab.org/competitions/19160#fn1

[5]For more details about UCCA semantic categories and the way they are used for the shared task, see https://competitions.codalab.org/competitions/19160#learn_the_details-overview. Our system does not predict remote edges defined in UCCA.

		labeled				unlabeled			
dataset	track	Avg. F1	P	R	F1	Avg. F1	P	R	F1
UCCA_English-Wiki	closed	0.657	0.673	0.655	0.664	0.809	0.829	0.807	0.818
UCCA_English-20K	closed	0.626	0.632	0.642	0.637	0.8	0.808	0.821	0.814
UCCA_German-20K	closed	0.71	0.72	0.72	0.72	0.851	0.863	0.862	0.862
UCCA_French-20K*	open	0.438	0.443	0.447	0.445	0.690	0.698	0.705	0.702

Table 1: Official results of the shared task evaluation for predicting different semantic category labels. (* The results for French are for Post-Evaluation.)

dataset	(D)	(C)	(N)	(E)	(F)	(G)	(L)	(H)	(A)	(P)	(U)	(R)	(S)	(Terminal)
English-Wiki	0.7	0.708	**0.866**	0.738	0.801	**0.286**	**0.836**	0.289	0.582	0.451	0.948	**0.914**	0	0.997
English-20K	0.521	0.733	**0.776**	0.743	0.647	0.04	**0.719**	0.248	0.538	0.527	0.978	**0.844**	0	0.997
German-20K	0.691	0.813	**0.796**	0.82	0.845	**0.778**	0.834	0.375	0.697	0.561	0.997	**0.916**	0	0.998
French-20K*	0.223	0.569	0.579	0.551	0.378	0.000	0.536	0.118	0.314	0.358	0.987	0.711	0	0.993

Table 2: Official results of the shared task evaluation for predicting Primary edges and their labels. (* The results for French are for Post-Evaluation.)

the reported results for our system can be misleading. In order to further investigate this issue, we performed an evaluation using the English Wiki development data, comparing the predicted labels with the gold standard in development set in the CONLLU format. The average F1-score for labelled edges was 0.71 compared to the 0.685 score our system achieved on the development set using the official evaluation script.

This clearly demonstrates that our system fares significantly better if it receives its input in the form of bi-lexical dependency graphs. Therefore, the system is best suited for semantic dependency parsing, although we believe that promising results could also be achieved in UCCA annotation if the conversion between the CONLLU and UCCA formats is improved to map and preserve information more accurately.

6 Conclusion and Future Work

In this paper, we described the system we submitted to the SemEval-2019 Task 1: 'Semantic Parsing using Graph Convolutional and Recurrent Neural Networks'. The model performs semantic parsing using information derived from syntactic dependencies between words in each sentence. We developed the model using a combination of GCN and BiLSTM components. Due to the penalisation resulting from the use of lossy CONLLU files, we argue that the results cannot be directly compared with those of the other task participants. [6]

In the future, we would like to build on the work presented in this paper by applying the architecture to the standard UCCA dataset, or possibly training the system to perform bi-lexical semantic dependency annotation.

[6]The code is available at https://github.com/shivaat/GCN-Sem.

References

Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *Proc. of ACL*, pages 228–238.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64, Brussels, Belgium. Association for Computational Linguistics.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750. Association for Computational Linguistics.

William Croft and D. A Cruse. 2004. *Cognitive linguistics*. Cambridge University Press.

Robert M. W Dixon. 2012. *Basic linguistic theory*. Oxford University Press.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490. Association for Computational Linguistics.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing prague czech-english dependency treebank 2.0. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pages 3153–3160. ELRA, European Language Resources Association.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for ucca. In *Proc. of ACL*, pages 1127–1138.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018. Multitask parsing across semantic representations. In *Proc. of ACL*, pages 373–385.

Daniel Hershcovich, Zohar Aizenbud, Leshem Choshen, Elior Sulem, Ari Rappoport, and Omri Abend. 2019. Semeval 2019 task 1: Cross-lingual semantic parsing with ucca.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia. Association for Computational Linguistics.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Glsen Eryigit, Sandra Kbler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02).

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Yi Zhang. 2014. Semeval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 63–72.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL*.

Omid Rohanian, Shiva Taslimipoor, Samaneh Kouchaki, Le An Ha, and Ruslan Mitkov. 2019. Bridging the gap: Attending to discontinuity in identification of multiword expressions.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks. *arXiv preprint arXiv:1703.06103*.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038. Association for Computational Linguistics.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015a. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015b. Grammar as a foreign language. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2773–2781. Curran Associates, Inc.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1127–1137. Association for Computational Linguistics.

MaskParse@Deskiñ at SemEval-2019 Task 1: Cross-lingual UCCA Semantic Parsing using Recursive Masked Sequence Tagging

Gabriel Marzinotto[1,2] **Johannes Heinecke**[1] **Géraldine Damnati**[1]
(1) Orange Labs / Lannion France
(2) Aix Marseille Univ, CNRS, LIS / Marseille France
{gabriel.marzinotto,johannes.heinecke,geraldine.damnati}@orange.com

Abstract

This paper describes our recursive system for SemEval-2019 *Task 1: Cross-lingual Semantic Parsing with UCCA*. Each recursive step consists of two parts. We first perform semantic parsing using a sequence tagger to estimate the probabilities of the UCCA categories in the sentence. Then, we apply a decoding policy which interprets these probabilities and builds the graph nodes. Parsing is done recursively, we perform a first inference on the sentence to extract the main scenes and links and then we recursively apply our model on the sentence using a masking feature that reflects the decisions made in previous steps. Process continues until the terminal nodes are reached. We choose a standard neural tagger and we focused on our recursive parsing strategy and on the cross lingual transfer problem to develop a robust model for the French language, using only few training samples.

1 Introduction

Semantic representation is an essential part of NLP. For this reason, several semantic representation paradigms have been proposed. Among them we find PropBank (Palmer et al., 2005) and FrameNet Semantics (Baker et al., 1998), Abstract Meaning Representation (AMR) (Banarescu et al., 2013), Universal Decompositional Semantics (White et al., 2016) and Universal Conceptual Cognitive Annotation (UCCA) (Abend and Rappoport, 2013). These constantly improving representations, along with the advances in semantic parsing, have proven to be beneficial in many NLU tasks such as Question Answering (Shen and Lapata, 2007), text summarization (Genest and Lapalme, 2011), dialog systems (Tur et al., 2005), information extraction (Bastianelli et al., 2013) and machine translation (Liu and Gildea, 2010).

UCCA is a cross-lingual semantic representation scheme, has demonstrated applicability in English, French and German (with pilot annotation projects on Czech, Russian and Hebrew). Despite the newness of UCCA, it has proven useful for defining semantic evaluation measures in text-to-text generation and machine translation (Birch et al., 2016). UCCA represents the semantics of a sentence using directed acyclic graphs (DAGs), where terminal nodes correspond to text tokens, and non-terminal nodes to higher level semantic units. Edges are labelled, indicating the role of a child in the relation to its parent. UCCA parsing is a recent task and since UCCA has several unique properties, adapting syntactic parsers or parsers from other semantic representations is not straight-forward. Current state of the art parser TUPA (Hershcovich et al., 2017) uses a transition based parsing to build UCCA representations.

Building over previous work on FrameNet Semantic Parsing (Marzinotto et al., 2018a,b) we chose to perform UCCA parsing using sequence tagging methods along with a graph decoding policy. To do this we propose a recursive strategy in which we perform a first inference on the sentence to extract the main scenes and links and then we recursively apply our model on the sentence with a masking mechanism at the input in order to feed information about the previous parsing decisions.

2 Model

Our system consists of a sequence tagger that is first applied on the sentence to extract the main scenes and links and then it is recursively applied on the extracted element to build the semantic graph. At each step of the recursion we use a masking mechanism to feed information about the previous stages into the model. In order to convert the sequence labels into nodes of the UCCA graph we also apply a decoding policy at each stage.

Our tagger is implemented using deep bidirectional GRU (*biGRU*). This simple architecture is frequently used in semantic parsers across different representation paradigms. Besides its flexibility, it is a powerful model, with close to state of the art performance on both PropBank (He et al., 2017) and FrameNet semantic parsing (Yang and Mitchell, 2017; Marzinotto et al., 2018b).

More precisely, the model consists of a 4 layer bi-directional Gated Recurrent Unit (GRU) with highway connections (Srivastava et al., 2015). Our model uses has a rich set of features including syntactic, morphological, lexical and surface features, which have shown to be useful in language abstracted representations. The list is given below:

- Word embeddings of 300 dimensions [1].
- Syntactic dependencies of each token[2].
- Part-of-speech and morphological features such as gender, number, voice and degree[2].
- Capitalization and word length encoding.
- Prefixes and Suffixes of 2 and 3 characters.
- A language indicator feature.
- Boolean indicator of idioms and multi word expression. Detailed in section 3.2.
- Masking mechanism, which indicates, for a given node in the graph, the tokens within the span as well as the arc label between the node and its parent. See details in section 2.1.

Except for words where we use pre-trained embeddings, we use randomly initialized embedding layers for categorical features.

2.1 Masking Mechanism

We introduce an original masking mechanism in order to feed information about the previous parsing stages into the model. During parsing, we first do an initial inference step to extract the main scenes and links. Then, for each resulting node, we build a new input which is essentially the same, but with a categorical sequence masking feature. For the input tokens in the node span, this feature is equal to the label of the arc between the node and its parent. Outside of the node span, this mask is equal to `O`. A diagram of this masking process is shown in figure 1. The process continues and the model recursively extracts the inner semantic structures (the node's children) in the graph, until the terminal nodes are reached.

[1]Obtained from https://github.com/facebookresearch/MUSE

[2] Using Universal Dependencies categories.

To train such a model, we build a new training corpus in which the sentences are repeated several times. More precisely, a sentence appears N times (N being the number of non terminal nodes in the UCCA graph) each one a with different mask.

2.2 Multi-Task UCCA Objective

Along with the UCCA-XML graph representations, a simplified tree representation in CoNLL format was also provided. Our model combines both representations using a multitask objective with two tasks. `TASK1` consists in, for a given node and its corresponding mask, predicting the children and their arc labels. `TASK1` encodes the children spans using a BIO scheme. The `TASK2` consists in predicting the CoNLL simplified UCCA structure of the sentence. More precisely, `TASK2` is a sequence tagger that predicts the UCCA-CoNLL function of each token. `TASK2` is not used for inference purposes. It is only a support that help the model to extract relevant features, allowing it to model the whole sentence even when parsing small pre-terminal nodes.

2.3 Label Encoding

We have previously stated that `TASK1` uses BIO encoded labels to model the structure of the children of each node in the semantic graph. In some rare cases, the BIO encoding scheme is not sufficient to model the interaction between parallel scenes. For example, when we have two parallel scenes and one of them appears as a clause inside the other. In such cases, BIO encoding does not allow to determine whether the last part of the sentence belongs to the first scene or to the clause. Despite this issue, prior experiments testing more complete label encoding schemes (BIEO, BIEOW) showed that BIO outperforms the other schemes on the validation sets.

2.4 Graph Decoding

During the decoding phase, we convert the BIO labels into graph nodes. To do so, we add a few constraints to ensure the outputs are feasible UCCA graphs that respect the sentence's structure:

- We merge parallel scenes (H) that do not have either a verb or an action noun to the nearest previous scene having one.
- Within each parallel scene, we force the existence of one and only one `State` (S) or `Process` (P) by selecting the token with the highest probability of `State` or `Process`.

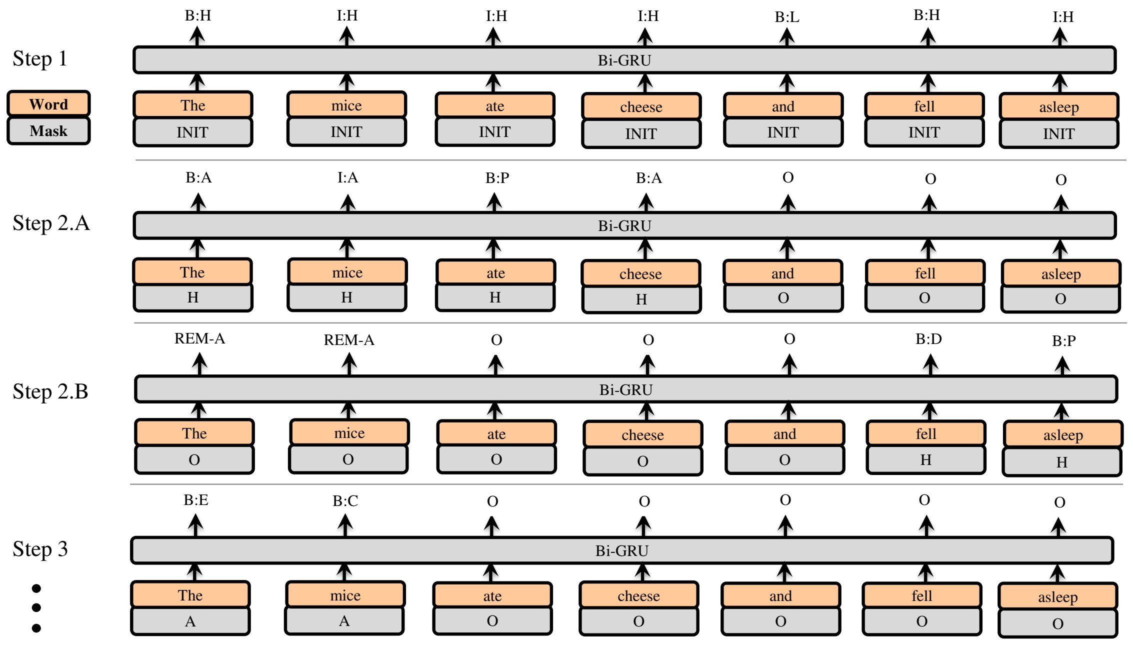

B:H I:H I:H I:H B:L B:H I:H
Step 1
Bi-GRU
Word
Mask
The mice ate cheese and fell asleep
INIT INIT INIT INIT INIT INIT INIT
B:A I:A B:P B:A O O O
Step 2.A
Bi-GRU
The mice ate cheese and fell asleep
H H H H O O O
REM-A REM-A O O O B:D B:P
Step 2.B
Bi-GRU
The mice ate cheese and fell asleep
O O O O O H H
B:E B:C O O O O O
Step 3
Bi-GRU
The mice ate cheese and fell asleep
A A O O O O O

Figure 1: Masking mechanism through recursive calls. `Step 1` parses the sentence to extract parallel scenes (H) and links (L). Then `Steps 2.A 2.B` use a different mask to parse these scenes and extract arguments (A) and processes (P) which will be recursively parsed until terminal nodes are reached.

- For scenes (H) and arguments (A) we do not allow to split multi word expressions (MWE) and chunks into different graph nodes. If the boundary between two segments lies inside a chunk or MWE segments are merged.

2.5 Remote Edges

Our approach easily handles remote edges. We consider remote arguments as those detected outside the parent's node span (see `REM` in Fig.1). Our earlier models showed low recall on remotes. To fix this, we introduced a detection threshold on the output probabilities. This increased the recall at the cost of some precision. The optimal detection threshold was optimized on the validation set.

3 Data

3.1 UCCA Task Data

In table 1 we show the number of annotations for each language and domain. Our objective is to build a model that generalizes to the French language despite of having only 15 training samples.

When we analyse data in details we observe that there are several tokenization errors. Specially in the French corpus. These errors propagate to the POS tagging and dependency parsing as well. For this reason, we retokenized and parsed all the corpus using a enriched version of UDpipe that we trained ourselves (Straka and Straková, 2017) using the Treebanks from Universal Dependencies[3]. For French we enriched the Treebank with XPOS from our lexicon. Finally, since tokenization is pre-established in the UCCA corpus we projected the improved POS and dependency parsing into the original tokenization of the task.

Corpus	Train	Dev	Test
English Wiki	4113	514	515
English 20K	-	-	492
German 20K	5211	651	652
French 20K	15	238	239

Table 1: number of UCCA annotated sentences in the partitions for each language and domain

3.2 Supplementary lexicon

We observed that a major difficulty in UCCA parsing is analyzing idioms and phrases. The unawareness about these expressions, which are mostly used as links between scenes, mislead the model during the early stages of the inference and errors get propagated through the graph. To boost the performance of our model when detecting links and parallel scenes we developed an internal list with about 500 expression for each language. These lists include prepositional, adverbial and conjunctive expressions and are used to compute Boolean features indicating the words in the sentence which are part of an expression.

[3] https://universaldependencies.org/

Open Tracks	Ours Labeled			Ours Unlabeled			TUPA Labeled			TUPA Unlabeled		
	Avg F1	Prim F1	Rem F1	Avg F1	Prim F1	Rem F1	Avg F1	Prim F1	Rem F1	Avg F1	Prim F1	Rem F1
Dev English Wiki	**70.8**	71.3	58.7	82.5	83.8	37.5	**74.8**	75.3	51.4	86.3	87.0	51.4
Dev German 20K	**74.7**	75.4	40.5	87.4	88.6	40.9	**79.2**	79.7	58.7	90.7	91.5	59.0
Dev French 20K	**<u>63.6</u>**	64.4	19.0	78.9	79.6	20.5	**<u>51.4</u>**	52.3	1.6	74.9	76.2	1.6
Test English Wiki	**68.9**	69.4	42.5	82.3	83.1	42.8	**73.5**	73.9	53.5	85.1	85.7	54.3
Test English 20K	**66.6**	67.7	24.6	82.0	83.4	24.9	**68.4**	69.4	25.9	82.5	83.9	26.2
Test German 20K	**74.2**	74.8	47.3	87.1	88.0	47.6	**79.1**	79.6	59.9	90.3	91.0	60.5
Test French 20K	**<u>65.4</u>**	66.6	24.3	80.9	82.5	25.8	**<u>48.7</u>**	49.6	2.4	74.0	75.3	3.2

Table 2: Our model vs TUPA baseline performance for each open track

Tracks	D	C	N	E	F	G	L	H	A	P	U	R	S
EN Wiki	64.3	71.4	68.5	69.6	76.7	0.0	71.4	61.3	60.0	64.0	99.7	89.2	25.1
EN 20K	47.2	75.2	62.5	72.3	71.5	0.2	57.9	49.5	55.7	69.8	99.7	83.2	19.5
DE 20K	69.4	83.8	57.7	80.5	83.8	59.2	68.4	62.2	67.5	68.9	97.1	86.9	25.9
FR 20K	46.1	76.0	58.9	71.2	53.3	4.8	59.4	50.4	52.8	67.6	99.6	83.5	16.9

Table 3: Our model's Fine-grained F1 by label on Test Open Tracks

3.3 Multilingual Training

This model uses multilingual word embeddings trained using fastText (Bojanowski et al., 2017) and aligned using MUSE (Conneau et al., 2017). This is done in order to ease cross-lingual training. In prior experiments we introduced an adversarial objective similar to (Kim et al., 2017; Marzinotto et al., 2019) to build a language independent representation. However, the language imbalance on the training data did not allow us to take advantage from this technique. Hence, we simply merged training data from different languages.

4 Experiments

We focus on obtaining the model that best generalizes on the French language. We trained our model for 50 epochs and we selected the best one on the validation set. In our experiments we did not use any product of experts or bagging technique and we did not run any hyper parameter optimization.

We trained several models building different training corpora composed of different language combinations. We obtained our best model using the training data for all the languages. This model `FR+DE+EN` achieved 63.6% avg. F1 on the French validation set. Compared to 63.1% for `FR+DE`, 62.9% for `FR+EN` and 50.8% for only `FR`.

4.1 Main Results

In Table 2 we provide the performance of our model for all the open tracks and we provide the results for TUPA baseline in order to establish a comparison. Our model finishes 4th in the French Open Track with an average F1 score of 65.4%, very close to the 3rd place which had a 65.6% F1. For languages with larger training corpus, our model did not outperform the monolingual TUPA.

4.2 Error Analysis

In Table 3 we give the performance by arc type. We observe that the main performance bottleneck is in the parallel scene segmentation (H). Due to our recursive parsing approach, this kind of error is particularly harmful to the model performance, because scene segmentation errors at the early steps of the parsing may induce errors in the rest of the graph. To assert this, we used the validation set to compare the performance of the mono scene sentences (with no potential scene segmentation problems) with the multi scene sentences. For the French track we obtained 67.2% avg. F1 on the 114 mono scene sentences compared to 61.9% avg. F1 on the 124 multi scene sentences.

5 Conclusions

We described an original approach to recursively build the UCCA semantic graph using a sequence tagger along with a masking mechanism and a decoding policy. Even though this approach did not yield the best results in the UCCA task, we believe that our original recursive, mask-based parsing can be helpful in low resource languages. Moreover, we believe that this model could be further improved by introducing a global criterion and by performing further hyper parameter tuning.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (UCCA). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 228–238.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Emanuele Bastianelli, Giuseppe Castellucci, Danilo Croce, and Roberto Basili. 2013. Textual inference and meaning representation in human robot interaction. In *Proceedings of the Joint Symposium on Semantic Processing. Textual Inference and Structures in Corpora*, pages 65–69.

Alexandra Birch, Omri Abend, Ondřej Bojar, and Barry Haddow. 2016. HUME: Human UCCA-based evaluation of machine translation. *arXiv preprint arXiv:1607.00030*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Pierre-Etienne Genest and Guy Lapalme. 2011. Framework for abstractive summarization using text-to-text generation. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, Portland, Oregon, USA. Association for Computational Linguistics, Association for Computational Linguistics.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. *arXiv preprint arXiv:1704.00552*.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2017. Cross-lingual transfer learning for pos tagging without cross-lingual resources. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2832–2838. Association for Computational Linguistics.

Ding Liu and Daniel Gildea. 2010. Semantic role features for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 716–724, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gabriel Marzinotto, Jeremy Auguste, Frédéric Béchet, Géraldine Damnati, and Alexis Nasr. 2018a. Semantic Frame Parsing for Information Extraction : the CALOR corpus. In *LREC 2018*, Miyazaki, Japan.

Gabriel Marzinotto, Frédéric Béchet, Géraldine Damnati, and Alexis Nasr. 2018b. Sources of Complexity in Semantic Frame Parsing for Information Extraction. In *International FrameNet Workshop 2018*, Miyazaki, Japan.

Gabriel Marzinotto, Géraldine Damnati, Frédéric Béchet, and Benoît Favre. 2019. Robust semantic parsing with adversarial learning for domain generalization. In *Proc. of NAACL*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 12–21, Prague. Association for Computational Linguistics, Association for Computational Linguistics.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *NIPS 2015*, pages 2377–2385, Montral, Qubec, Canada.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Gokhan Tur, Dilek Hakkani-Tür, and Ananlada Chotimongkol. 2005. Semi-supervised learning for spoken language understanding semantic role labeling. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 232 – 237.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on universal dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1713–1723.

Bishan Yang and Tom Mitchell. 2017. A joint sequential and relational model for frame-semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256. Association for Computational Linguistics.

Tüpa at SemEval-2019 Task 1: (Almost) feature-free Semantic Parsing

Tobias Pütz
Department of Linguistics
University of Tübingen
SFB 833 A3

Kevin Glocker
Department of Linguistics
University of Tübingen

{tobias.puetz,kevin.glocker}@student.uni-tuebingen.de

Abstract

Our submission for Task 1 'Cross-lingual Semantic Parsing with UCCA' at SemEval-2018 is a feed-forward neural network that builds upon an existing state-of-the-art transition-based directed acyclic graph parser. We replace most of its features by deep contextualized word embeddings and introduce an approximation to represent non-terminal nodes in the graph as an aggregation of their terminal children. We further demonstrate how augmenting data using the baseline systems provides a consistent advantage in all open submission tracks. We submitted results to all open tracks (English, in- and out-of-domain, German in-domain and French in-domain, low-resource). Our system achieves competitive performance in all settings besides the French, where we did not augment the data. Post-evaluation experiments showed that data augmentation is especially crucial in this setting.

1 Introduction

Semantic Parsing is the task of assigning an utterance a structured representation of its meaning. The goal is to assign similar structures to utterances with similar meanings, regardless of their syntactic realizations. In Syntactic Parsing, for instance, the sentence 'John saw Paul.' will have a different structure than 'Paul was seen by John'. Semantic Parsing, in contrast, aims to solely encode the fact that John saw Paul. Deriving a semantic representation of an utterance has various applications. It can serve as a starting point for the evaluation of machine translation systems, as the structure of the semantic representation should be similar across languages. Birch et al. (2016) use human annotated scores of individual UCCA semantic units in their HUME metric to provide a fine-grained analysis of translation quality and improve scalability to longer sentences by approximating human judgement semi-automatically from the annotated scores of each unit. Explicit semantic representations could also provide the structured information necessary to alleviate recent issues in Natural Language Inference (NLI) where McCoy and Linzen (2019) showed that state-of-the-art NLI systems fail to recognize that e.g. 'Alice believes Mary is lying.' does not entail 'Alice believes Mary.'. Using precise semantic representations of the sentences a theorem could be built on which various logical inferences can be performed with a theorem prover such as in Martínez-Gómez et al. (2016).

Universal Conceptual Cognitive Annotation (UCCA) (Abend and Rappoport, 2013) is a semantic grammar formalism where natural language expressions are analyzed as deep directed acyclic graph (DAG) structures, deep meaning the graphs feature non-terminal nodes. Due to it's coarse-grained representation using cognitively motivated categories it is both domain and language independent and quickly learned even by annotators without a linguistic background (Abend and Rappoport, 2013).

The goal of the SemEval-2018 Task 1 'Cross-lingual Semantic Parsing with UCCA' was to develop a parser producing UCCA-DAG structures trained on articles from Wikipedia in English and passages from the book "Twenty Thousand Leagues Under the Sea" in French and German. The parsers were evaluated on the DAG-F1 metric on in-domain passages in English, French and German as well as out-of-domain passages in English in both an open and a closed track (Hershcovich et al., 2018b). Since we made extensive use of external resources we participated only in the open track of all settings.

For our participation, we build upon the transition-based DAG parser Tupa (Hershcovich et al., 2017). Our adaptation reuses the transition system and oracle. We extend Tupa with respect to its representations of non-terminal nodes in a way that they are an aggregation of all their terminal nodes. While Tupa uses a Recurrent Neural Network, our system is a simple feed-forward network that uses a small set of features and ELMo contextualized embeddings (Peters et al., 2018) made available by Che et al. (2018)[1] and Fares et al. (2017).

2 Background

Until recently, semantic parsers were exclusively symbolic rule-based systems (Bos, 2005). These systems rely on complex hand-written and necessarily language-specific sets of rules, requiring a re-implementation for every new language. More recently, neural methods have also arrived in the domain of Semantic Parsing. They achieve state-of-the-art results while being largely language-agnostic. Since these systems usually require large amounts of annotated data, this line of work is largely concerned with the augmentation of training data. Hershcovich et al. (2018a) recognize the similarity between several annotation schemes and jointly learn to parse other semantic formalisms in a multi-task setting, while van Noord et al. (2018) add large amounts of automatically annotated data to their training data. Both approaches led to significant improvements over not using the additional data.

3 Silver Data

We created additional training data for both English and German using the open track baseline systems. The English silver data was taken from the 1B word benchmark (Chelba et al., 2014), the German from the archive of the newspaper taz. For both languages, we took the first 15,000 sentences of the corpora and added UCCA annotation using the baseline systems. Our training datasets then consisted of the concatenation of gold and silver data, and another gold only set. Due to a lack of time we did not create silver data for our French submission. Post-evaluation results for French, trained on v2.0 of the GSD treebank[2] provided by Universal Dependencies (Nivre et al., 2016), are presented in Section 6.1.

[1]https://github.com/HIT-SCIR/ELMoForManyLangs/

[2]https://universaldependencies.org/treebanks/fr_gsd/

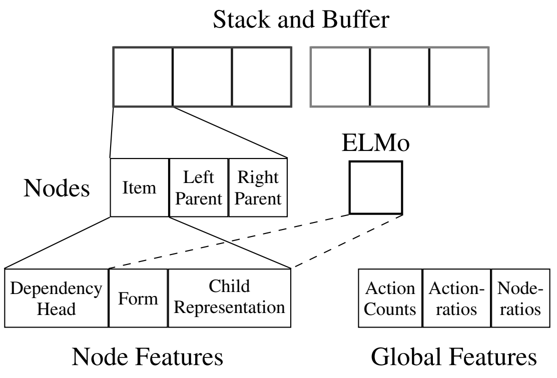


Figure 1: Illustration of the features used by Tüpa. The final feature vector results from the concatenation of all stack and buffer features with the global features. Features dropped after preliminary experiments are omitted for brevity.

4 System

Our system is an ensemble of small feed-forward neural networks. We use three global features: typed absolute counts for previous parser actions and action- and node-ratios (Hershcovich et al., 2017). We further follow the standard in transition-based parsing and extract a set of features based on the top three items on stack and buffer. To capture some of the structure of the partially built graph, we extract the rightmost and leftmost parents and children of the respective items, following Hershcovich et al. (2017). Each of these items is represented by the ELMo embedding of its form, the embedding of its dependency head and the embeddings of all terminal children. We use the average over all ELMo layers to retrieve the embedding of a word. Non-terminal nodes do not have a form or dependency head, hence these are represented by a learned non-terminal embedding. Both the non-terminals and terminals have a third feature, a representation of their children. In the case of terminals, this feature is equal to its form feature. For the non-terminals, it is an aggregation of all its children, produced by the child representation module. Figure 1 illustrates the set of features used by our system. We experimented with richer feature sets, including the last parser actions, named-entity, part-of-speech and dependency types, but dropped them after performing preliminary experiments. The input to the feed-forward module is the concatenation of all features with the output of the child representation module. The classification portion of the system was implemented using Tensorflow (Abadi et al., 2015).

4.1 Representing Non-Terminals

The child representation module aims to enrich the representation of non-terminal nodes. Our initial representation for non-terminal nodes was a set of discrete features describing the number of typed in- and outgoing edges and the nodes' height in the tree. While this might be informative on an abstract level, it does not provide any information about the content covered by this node. We solve this poverty of information by concatenating each of the embeddings of the terminal children of a node with an embedding for the first edge type leading to them. The resulting combination is fed through a dense layer with d neurons, resulting in n vectors with d dimensions where n is the number of terminals under the node. We then reduce the n vectors into a single d dimensional vector by taking the maximum value of each dimension. Figure 2 depicts how the representation of a non-terminal node is obtained. While it would be desirable to process the children using context-aware methods, such as RNNs or self attention, it is not feasible since some of the nodes can have more than 100 children. Future work should explore recursive formulations for representing a node by its direct children instead of relying on all terminal children, performing largely redundant operations for higher nodes.

4.2 Hyperparameters

We apply dropout (Srivastava et al., 2014) with a keep probability of 0.8 to the inputs of all layers. The child processing module is a single layer feed-forward network with 256 hidden units. The feed-forward module is single layer feed-forward network with 512 hidden units. Both modules use the ReLU activation function. Training is performed with the Adam optimizer (Kingma and Ba, 2014) using an initial learning rate of $8\mathrm{e}{-5}$ that is halved every two epochs without an improvement on development accuracy. We halt the training after five epochs without an improvement on development transition accuracy. The models were first trained on the concatenation of the silver and gold data and following the early stopping another time only on the gold data using the same parameters. We use mini-batches of size 192 and evaluate on the development set every 1000 mini-batches. As training time imposes a serious limitation, we did not perform an extensive hyperparameter search and settled on these after initial experiments.

	DAG-F1	Primary F1	Remote F1	Tupa-DAG
English Wiki	0.735	0.741	0.425	0.735
English 20k	0.709	0.719	0.296	0.684
German 20k	0.781	0.788	0.408	0.791
French 20k	0.456	0.464	0	0.487

Table 1: DAG-F1, primary F1 and remote F1 scores with the DAG-F1 score of the baseline on the test sets in the open tracks.

5 Results and Discussion

Table 1 shows the submission scores of our parser trained using the hyperparameters described in Section 4.2 on the test datasets in the open tracks. Since only 15 French passages were available for training, our French results were obtained by first training a model on the concatenation of the French passages and the German 20k training dataset using French ELMo embeddings. After convergence, it was fine-tuned on only the French passages for two epochs. However, this did not provide a significant increase in F1 score over a model trained exclusively on the French passages. All results were produced using a five model ensemble, consisting of the model with the best transition accuracy and the four following it before early stopping. The results show that our parser achieves competitive performance to the baseline while relying on fewer features. In particular, for the English in-domain data, we achieve the same performance as the baseline, for the out-of-domain data we surpass it by 0.025 DAG-F1. In German and French where only in-domain data exists our approach is outperformed by the baseline which we partially attribute to issues in the creation of the silver data. Post-submission results obtained after performing a more exhaustive hyperparameter search on the development set and with correct silver-data surpass the baseline performance on the test sets in all open settings.

6 Further Experiments

In this section, we will describe the findings of our post-evaluation experiments. We evaluated the effect of silver data and provide results for French with silver data. We further performed experiments on non-terminal representations and investigated the effect of model size. Since this only covers a fraction of our experiments and describing them all would be out of scope, we

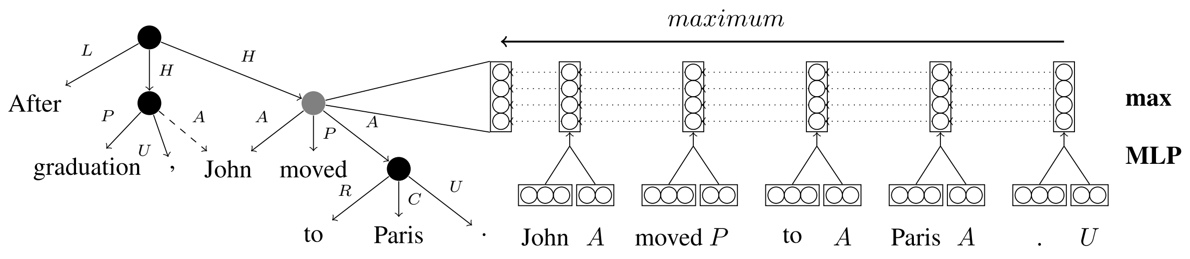


Figure 2: Depiction of a non-terminal representation. The terminal children dominated by the grey node are concatenated with the first edge leading to them and fed through a fully-connected layer. The multiple resulting vectors are reduced into a single one by taking the maximum value of each dimension.

	Full	**Submission**	**Forms only**
Gold	0.724	0.733	0.679
Silver+Gold	**0.739**	**0.744**	**0.688**

Table 2: DAG F1 scores on the English development set after training with gold and gold+silver data. Silver data provides a boost for all combinations.

Data	**avg. F1**	**remote F1**
Gold	0.456	0.0
Silver+Gold	**0.557**	**0.025**

Table 3: DAG F1 Scores on the French test set with and without silver data. Here in the low-resource setting, the effect of additional data is the largest. Without silver data, the parser did not predict any remote edges correctly.

	Full	**Submission**
Discrete	0.723	0.688
Aggregated	**0.739**	**0.744**

Table 4: Effect of discrete and aggregated non-terminal representations on the DAG F1 score on the English development set. The aggregated representation provides a clear advantage over the discrete one.

provide the full results alongside their hyperparameters at https://twuebi.github.io/publications/ucca_post_eval.pdf.

6.1 Silver Data

We measured the effect of silver data on English and French by evaluating several model configurations in two settings. The first setting matches the training data used for the submission and is the concatenation of the gold and silver data. In the second setting, the only available data is the gold data.

English: We trained three models for English. The first model configuration uses all features and corresponds to the model described in the end of Section 4, the second is our submission, described in Section 4. The last model uses only embeddings of the forms and dependency heads. As shown in Table 2, additional training data provides a consistent boost in F1 score across all tested feature combinations. Moreover, it seems that there is a larger effect of the silver data on models with more features, indicating a better estimation of the feature representations based on the additional data.

Low resource setting: Table 3 demonstrates the effect of silver data on French for the submission model configuration. The effect of additional data is the largest in the low-resource setting, providing a boost of 0.1 in average F1 score. Adding the silver data also leads to some of the remote edges being correct, whereas there are no correct remote edges for the gold-only model.

6.2 Non-Terminal Representation

To measure the effectiveness of our non-terminal representation, we ran two experiments using silver and gold data. In both cases, we trained one model with aggregated non-terminal representations and one with the discrete representations of typed in- and outgoing edges and the nodes' heights in the tree. The first experiment used all available features. The second was trained with the features of our submission. Table 4 presents the results of the experiments. The explicit child representations provide a clear improvement over the discrete representation. In the second experiment, where no in- and outgoing edges were used and the only non-terminal representations are the left- and rightmost children, the gap increased even further, in fact it is the worst F1 score of all models trained on silver data.

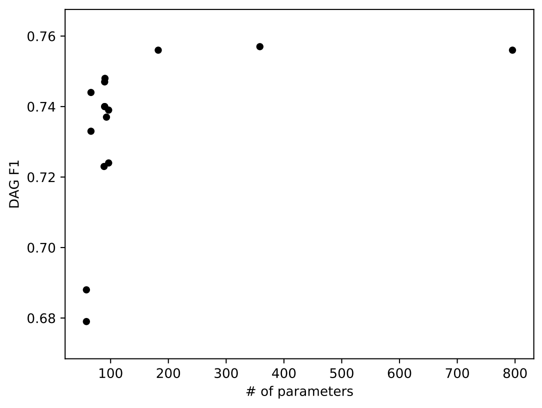


Figure 3: DAG F1 scores on English development data on the y-axis. Million parameters in the models on the x-axis. Larger models seem provide some improvements that begin to level off for big models.

6.3 Bigger means better?

Figure 3 contrasts the number of trainable parameters of the models in our experiments with the F1 score on the English development set. While there are some improvements for larger models, it can be seen that the effect begins to level off at 200M parameters and eventually leads to a small regression with the largest model. Possible causes are overfitting and a lack of training data. Future work should explore whether additional training data allows for larger models. Additional regularization such as L2 regularization might also prove useful. For our experiments, this was out of scope since training so many models was not feasible.

7 Conclusion

In this work, we presented a parser for the semantic grammar formalism UCCA. Our parser relies on a small set of features and achieves competitive performance on the English and German data, but lags behind on French where almost no training data is available. We demonstrated, using ablation experiments, that the explicit representation of non-terminals and additional silver data are crucial for our result. We have further shown that silver data is especially helpful in the low-resource setting where it boosts the average F1 score from 0.456 to 0.557. Future work should investigate how much more improvement additional data can provide. This should be explored both in form of other formalisms (Hershcovich et al., 2018a) and silver data (van Noord et al., 2018). Besides the data aspect, we also believe that improving the non-terminal representation will lead to significant gains. The goal should be to find a representation that leverages the recursive structure of the partially built graph.

Acknowledgements

We would like to thank Çağrı Çöltekin for his extensive comments on an earlier version of this work.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2015. Tensorflow: Large-scale machine learning on heterogeneous distributed systems.

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (UCCA). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 228–238.

Alexandra Birch, Barry Haddow, Ondrej Bojar, and Omri Abend. 2016. HUME: human UCCA-based evaluation of machine translation. *CoRR*, abs/1607.00030.

Johan Bos. 2005. Towards wide-coverage semantic interpretation.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64, Brussels, Belgium. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Murhaf Fares, Andrey Kutuzov, Stephan Oepen, and Erik Velldal. 2017. Word vectors, reuse, and replicability: Towards a community repository of large-text resources. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 271–276, Gothenburg, Sweden. Association for Computational Linguistics.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proc. of ACL*, pages 1127–1138.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018a. Multitask parsing across semantic representations. In *Proc. of ACL*, pages 373–385.

Daniel Hershcovich, Leshem Choshen, Elior Sulem, Zohar Aizenbud, Ari Rappoport, and Omri Abend. 2018b. Semeval 2019 shared task: Cross-lingual semantic parsing with ucca-call for participation. *arXiv preprint arXiv:1805.12386*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. 2016. ccg2lambda: A compositional semantics system. In *Proceedings of ACL 2016 System Demonstrations*, pages 85–90, Berlin, Germany. Association for Computational Linguistics.

Richard T McCoy and Tal Linzen. 2019. Non-entailed subsequences as a challenge for natural language inference. *Proceedings of the Society for Computation in Linguistics*, 2(1):358–360.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 23–28.

Rik van Noord, Lasha Abzianidze, Antonio Toral, and Johan Bos. 2018. Exploring neural methods for parsing discourse representation structures. *arXiv preprint arXiv:1810.12579*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

UC Davis at SemEval-2019 Task 1: DAG Semantic Parsing with Attention-based Decoder

Dian Yu
University of California, Davis
dianyu@ucdavis.edu

Kenji Sagae
University of California, Davis
sagae@ucdavis.edu

Abstract

We present a simple and accurate model for semantic parsing with UCCA as our submission for SemEval 2019 Task 1. We propose an encoder-decoder model that maps strings to directed acyclic graphs. Unlike many transition-based approaches, our approach does not use a state representation, and unlike graph-based parsers, it does not score graphs directly. Instead, we encode input sentences with a bidirectional-LSTM, and decode with self-attention to build a graph structure. Results show that our parser is simple and effective for semantic parsing with reentrancy and discontinuous structures.

1 Introduction

Semantic parsing aims to capture structural relationships between input strings and graph representations of sentence meaning, going beyond concerns of surface word order, phrases and relationships. The focus on meaning rather than surface relations often requires the use of reentrant nodes and discontinuous structures. Universal Conceptual Cognitive Annotation (UCCA) (Abend and Rappoport, 2013) is designed to support semantic parsing with mappings between sentences and their corresponding meanings in a framework intended to be applicable across languages.

SemEval 2019 Task 1 (Hershcovich et al., 2018b, 2019) focuses on semantic parsing of texts into graphs consisting of terminal nodes that represent words, non-terminal nodes that represent internal structure, and labeled edges representing relationships between nodes (e.g. *participant*, *center*, *linker*, *adverbial*, *elaborator*), according to the UCCA scheme. Annotated datasets are provided, and participants are evaluated in four settings: English with domain-specific data, English with out-of-domain data, German with domain-specific data, and French with only development and test data, but no training data. Additionally, there are open and closed tracks, where the use of additional resources is and is not allowed, respectively. Our entry in the task is limited to the closed track and the first setting, domain-specific English using the Wiki corpus, where the relatively small dataset (4113 sentences for training, 514 for development, and 515 for testing) consists of annotated sentences from English Wikipedia.

Our model follows the encoder-decoder architecture commonly used in state-of-the-art neural parsing models (Kitaev and Klein, 2018; Kiperwasser and Goldberg, 2016b; Cross and Huang, 2016; Chen and Manning, 2014). However, we propose a very simple decoder architecture that relies only on a recursive attention mechanism of the encoded latent representation. In other words, the decoder does not require state encoding and model-optimal inference whatsoever. Our novel model achieved a macro-averaged F1-score of 0.753 in labeled primary edges and 0.864 in unlabeled primary edge prediction on the test set. The results confirm the suitability of our proposed model to the semantic parsing task.

2 Related work

Leveraging parallels between UCCA and known approaches for syntactic parsing, Hershcovich et al. (2017) proposed TUPA, a customized transition-based parser with dense feature representation. Based on this model, Hershcovich et al. (2018a) used multitask learning effectively by training a UCCA model along with similar parsing tasks where more training data is available, such as Abstract Meaning Representation (AMR) (Banarescu et al., 2013) and Universal Dependencies (UD) (Nivre et al., 2016). Due to

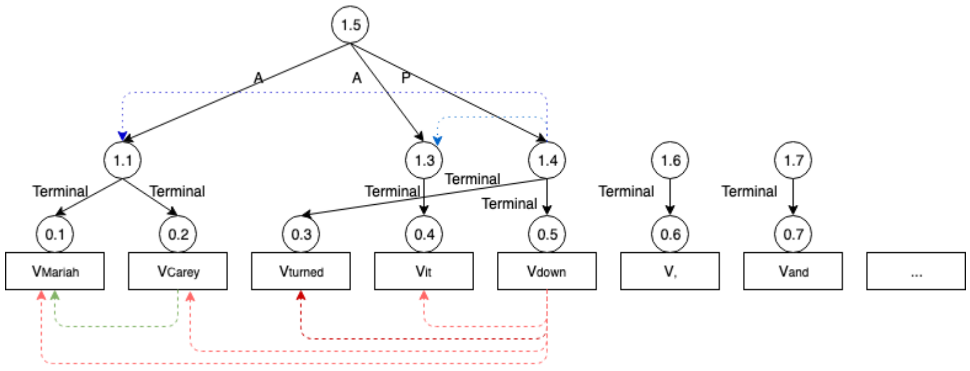


Figure 1: Illustration of the decoder for the beginning of a sentence, "Mariah Carey turned it down, and ...". Each v_i represents the context embedding for each word i from the BiLSTM encoder. Words on edges represent category labels between nodes, where A is participant and P is process. Circles represent nodes in the graph, each with a pair in indices. Circles with 0 as the first index are terminal nodes, and circles with 1 as the first index are non-terminal nodes. (1). Dashed green lines represent the attention mechanism for the word $Carey$, which forms a continuous proper noun "Mariah Carey". (2). Dashed red lines represent the attention mechanism for the word $down$, which forms a discontinuous unit "turned ... down". (3). Dotted blue lines represent the attention mechanism for $node_{1.4}$. The darker the color, the higher the attention score.

the requirements of reentrancy, discontinuity, and non-terminals, other powerful parsers were shown to be less suitable for parsing with UCCA (Hershcovich et al., 2017).

3 Parsing Model

BiLSTM models are capable of providing feature representations with sequential data, and attention mechanisms (Vaswani et al., 2017) have been applied successfully to parsing tasks (Kitaev and Klein, 2018). Inspired by their success, our model uses a BiLSTM encoder and a self-attention decoder. The encoder represents each node (terminal and non-terminal) in the DAG without the need to encode features and the current parser state. The proposed decoder takes the encoded representation as the configuration and uses attention mechanism. Without any additional feature extraction, it serves a similar role as an oracle and a transition-system in transition-based parsers. We jointly train a label prediction model and a discontinuity prediction model. We predict remote edges with a different encoder. An example of the parsing model can be seen in Figure 1.

3.1 Terminal Nodes

To mitigate sparsity due to the small amount of training data available, we concatenate part-of-speech tags embeddings to word embeddings in terminal nodes. In addition, because the connections between terminal nodes and non-terminal nodes often require identification of named entities, we also added entity type and case information as additional knowledge. Given a sentence $\mathbf{x} = x_1, ..., x_n$, the vector for each input token is thus represented as $u_i = emb(x_i) \circ emb(pos_i) \circ emb(entity\_type_i) \circ emb(case_i)$, where $case_i$ is 1 if the first character of the word is capitalized and 0 otherwise. We use pretrained word embeddings from fastText[1] for $emb(x_i)$. POS tags and entity types are predicted using external models[2] and are provided in the training corpus. Each word representation from the encoder is $v_i = BiLSTM(u_i)$. We assign these contextual word embeddings as vectors to terminal nodes.

3.2 Non-terminal Nodes

For non-terminal nodes with only one terminal node as the child, the representation is the same as its corresponding terminal node, i.e. a contextual word embedding from the BiLSTM encoder. For other non-terminal nodes that have more than one terminal children or non-terminal children (i.e. represent more than one word in the text), we use a span representation. Following Cross and Huang (2016), we represent the span between the words x_i, x_j as $v_{i,j} = (f_j - f_i) \circ (b_i - b_j)$ where $f_0, ..., f_n$ and $b_0, ..., b_n$ are the output of the forward and backward directions in the BiLSTM, respectively. However, the linear subtractions from a nonlinear recurrent neural network (RNN) as a span approximation is not intuitive. Instead, we experimented

[1] https://fasttext.cc/
[2] https://spacy.io/

with an additional BiLSTM on the target span $x_i, x_{i+1}, ..., x_j$, similar to the recursive tree representations in (Socher et al., 2013; Kiperwasser and Goldberg, 2016a) but replaced the feed-forward network with an LSTM. In our experiments with the small dataset in the closed track of the English domain-specific track, this method did not result in improved performance.

3.3 Attention Mechanism For Decoding

Our basic decoding model is inspired by the global attention mechanism used in machine translation. The attention averages the encoded state in each time step in the sequence with trainable weights (Luong et al., 2015). We set a maximum sequence length and calculate the attention weights (in probability) for the left boundary index of the span given the node representation $v_{i,j}$ $(i \leq j)$:

$$h_{span} = MLP(v_{i,j}) \tag{1}$$

$$p_{left\_boundary} = softmax(h_{span}) \tag{2}$$

where MLP is a multilayer perceptron and h_{span} is of size (1, max_sequence_length). We choose $\arg\max_i p_{left\_boundary}$ as the index of the left boundary of the predicted span. Let j_l denote the index of the left most child of the node j (for example, in Figure 1, j_l for $node_{1.5}$ is 1 and j_l for $node_{1.6}$ is 6)[3]. If $i \geq j_l$, then the node attends to itself to indicate that a span cannot be created yet (as is the case for $node_{1.6}$ in Figure 1). Otherwise, there is a span that forms a semantic unit and we need to create a parent node. For example, $i = 1$ for the $node_{1.4}$, so we create a new $node_{1.5}$ which connects the nodes within the span [1: 5], i.e. $node_{1.1}$, $node_{1.3}$, and $node_{1.4}$.

We do this recursively to attend to a previous index until the node attends to itself. Then we repeat the procedure on the next word in the sequence. The illustration is shown in Figure 1 with dotted blue lines. The algorithm is presented in Algorithm 1 below. $primary\_parent$ indicates the parent node to which the current node is not a remote child (in the DAG setting, a child node may have multiple parents). We set the maximum number of recurrence to be 7 to prevent excessive node creation during inference.

Despite its simplicity, there are two limitations to this method. One is the restriction of the maximum sequence length. The other is the distinction

[3]For simplicity, word indices start at 1 in the Figure.

Algorithm 1 Index-attention decoder

```
1: for recur_num = 1 to max_recur do
2:     if $i \geq j_l$ then
3:         break
4:     end if
5:     $h_{span} = MLP(v_{i,j})$
6:     $i_{attn} = \underset{i}{\text{argmax}}\ softmax(h_{span})$
7:     $i = primary\_parent(v_{i_{attn}})_l$
8: end for
```

between the indices and the actual words in each sentence. The model may cheat during training to attend to specific indices regardless of the actual words in these indices.

Motivated by the success of biaffine attention(Dozat and Manning, 2016) and self-attention models (Vaswani et al., 2017), we replace the index attention decoder with a multiplication model where we can leverage fast optimized matrix multiplication. Similar to the left most child, let j_r denote the index of the right most child of $node_j$. $v_o = v[1 : j_r]$ where v is the output from the encoder of size (sequence_length, batch_size, hidden_size). The scoring function is defined as:

$$h_i = ReLU(W \times v_i + b) \tag{3}$$

$$h_o = ReLU(W \times v_o + b) \tag{4}$$

$$mm = matrix\_multiplication(h_i, h_o^T) \tag{5}$$

$$p_{left\_boundary} = softmax(mm) \tag{6}$$

Compared to the index attention decoder above, this decoder considers both the index and the span representation and thus is more flexible and robust to new texts. The recurrence call remains the same by replacing line 5 and 6 in Algorithm 1 with equations 3 − 6.

3.4 Label Prediction

Contextual information is important to label prediction. For instance, in the sentence "It announced Carey returned to the studio to start ... ", the phrase "Carey returned to the studio" should be labeled as a participant (A) instead of a scene (H) according to the context. Ideally the encoder will capture the information from the whole sentence so that we only need the current span to predict its label (since the span has the context information from both sides). However, as shown in previous research with RNN models, the contextual information is lost for a relatively long sentence. Therefore, similar to the label prediction problem with dependency parsers, we use a MLP to predict the label of a span $v_{i,j}$ given its context $p = primary\_parent(v_{i,j})$.

$$h = ReLU(W_l^1 \times (p \circ v_{i,j}) + b_l^1) \quad (7)$$

$$l = \underset{l}{argmax}\ softmax(W_l^2 * h + b_l^2) \quad (8)$$

We also experimented with only using span representation as seen in constituency parsing (Gaddy et al., 2018) by replacing $(p \circ v_{i,j})$ with $v_{i,j}$ in equation 7. Surprisingly, this increased the F1 score on the development set by 1.4 points. We conjecture that this is due to the limited amount of training data, which makes it more difficult to learn noisier representations.

3.5 Discontinuous Unit

After finding the left boundary of the current span unit as shown in section 3.3, we use two MLPs for binary classification to check (1) if the span forms a proper noun with which we need to combine multiple terminal nodes to one non-terminal node (as "Mariah Carey" in Figure 1) and (2) if the span forms a discontinuous unit (such as "turn ... down" in Figure 1).

$$prob_{propn} = W_p^2 \times ReLU(W_p^1 \times v_{i,j} + b_p^1) + b_p^2 \quad (9)$$

$$prob_{discont} = W_d^2 \times ReLU(W_d^1 \times v_{i,j} + b_d^1) + b_d^2 \quad (10)$$

If the node span attends to a node in the left and the model predicts a proper noun, we will create a non-terminal node and links all the terminal nodes $i, i+1, ..., j$ as its terminal children (shown as dashed green lines in Figure 1).

If the model predicts that the span is a discontinuous unit, instead of connecting all the terminal nodes as its children, the new created node only connects $node_i$ and $node_j$, and do the recurrence checks afterwards as shown in Algorithm 1 (illustrated as dashed red lines in Figure 1).

3.6 Remote Edges

We predict remote edges the same way as the matrix multiplication decoder for primary edges. We use a different BiLSTM encoder to learn representations and avoid confusion between attention to primary edges and remote edges.

	unlabeled(F1)	labeled(F1)
official	0.746	0.866
+ max_recur = 7	0.747	0.867
+ child_pred	0.760	0.87
+ $\beta_2 = 0.9$	0.762	0.87
+ bug_fix	**0.769**	**0.873**

Table 1: F1 score on primary edges evaluated on the development set

4 Training and Inference

During training, $node_i$ attends to the left most child of its primary parent ($node_p$) recursively until $node_p$ is not the left most child of $node_p$'s parent. Because a span representation contains information from both left to right and right to left, $node_i$ with the highest attention score not only contains the embedding of its terminal node, but also the span between index i and j in the text. We use cross entropy loss to jointly train for embeddings, the BiLSTM encoder, and the decoder.

For inference, we take the output of each token in the text from the BiLSTM encoder as input and create a non-terminal node for each terminal node. We create a new node when the token embedding attends to a different token outside of the current span boundary. The recurrence algorithm for each newly created non-terminal node shown in Algorithm 1 is applied.

5 Experiments

For the encoder, we use a 2-layer, 500 dimensional BiLSTM with 0.2 dropout. The word embedding size is 300 with feature embedding size of 20 each (pos tagging, entity type, and case information). We use Adam optimizer (Kingma and Ba, 2014) with β_2 set to 0.9 as suggested by Dozat and Manning (2016). Development set is used for early stopping. Because of the small dataset (4113 training sentences), the model overfits after 4 epochs.

6 Results

Table 1 provides the results on the development set and Table 2 shows the results on the test set. *official* shows results of the model we submitted to the competition with a maximum recursion number of 5 and a $\beta_2 = 0.99$. We obtained higher scores by increasing the recursion limit as in section 3.3 (+ max_revur = 7), using current span only

	primary		remote	
	unlabeled	labeled	unlabeled	labeled
baseline	0.733	0.858	**0.472**	**0.484**
official	0.73	**0.864**	-	-
final	**0.753**	**0.864**	0.447	0.447

Table 2: F1 score on primary and remote edges reported on the test set

as explained in section 3.4 (+ child_pred), changing β_2 as shown in section 5 (+ $\beta_2 = 0.9$) and fixing minor bugs (+ bug_fix) incrementally. *baseline* shows the results of the baseline model (TUPA) from Hershcovich et al. (2017). *final* shows the results of the model fine-tuned on the development set mentioned in Table 1.

Since there are normally 0 or 1 remote edges in each sentence in the training corpus, the remote edge prediction model is not as effective. Still, the model captures some remote relations. For example, in the sentence "Additionally, Carey's newly slimmed Figure began to change, as she stopped her exercise routines and gained weight", the node "gained weight" is predicted to point to "Carey" where the target annotated remote child is "she". Discontinuous unit prediction also suffers from the problem of insufficient training samples.

7 Conclusion

This paper describes the system that the UC Davis team submitted to SemEval 2019 Task 1. We propose a recursive self-attention decoder with a simple architecture. Our model is effective in UCCA semantic parsing, ranking third in the close track in-domain task with modest fine-tuning, highlighting the suitability of our approach.

Acknowledgments

This work was supported by the National Science Foundation under Grant No. 1840191. Any opinions, findings, and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of the NSF.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (ucca). In *ACL (1)*, pages 228–238. The Association for Computer Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750. ACL.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. *CoRR*, abs/1612.06475.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. What's going on in neural constituency parsers? an analysis. *CoRR*, abs/1804.07853.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1127–1138.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018a. Multitask parsing across semantic representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 373–385.

Daniel Hershcovich, Zohar Aizenbud, Leshem Choshen, Elior Sulem, Ari Rappoport, and Omri Abend. 2019. Semeval 2019 task 1: Cross-lingual semantic parsing with UCCA. *CoRR*, abs/1903.02953.

Daniel Hershcovich, Leshem Choshen, Elior Sulem, Zohar Aizenbud, Ari Rappoport, and Omri Abend. 2018b. Semeval 2019 shared task: Cross-lingual semantic parsing with UCCA - call for participation. *CoRR*, abs/1805.12386.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Eliyahu Kiperwasser and Yoav Goldberg. 2016a. Easy-first dependency parsing with hierarchical tree lstms. *CoRR*, abs/1603.00375.

Eliyahu Kiperwasser and Yoav Goldberg. 2016b. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *CoRR*, abs/1603.04351.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. *CoRR*, abs/1805.01052.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan T. McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016.*

R Socher, A Perelygin, J.Y. Wu, J Chuang, C.D. Manning, A.Y. Ng, and C Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. *EMNLP*, 1631:1631–1642.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

HHMM at SemEval-2019 Task 2: Unsupervised Frame Induction using Contextualized Word Embeddings

Saba Anwar[⋆], **Dmitry Ustalov**[†], **Nikolay Arefyev**[‡, §], **Simone Paolo Ponzetto**[†], **Chris Biemann**[⋆], and **Alexander Panchenko**[⋆,⋄]

[⋆]Language Technology Group, Department of Informatics, University of Hamburg, Germany
[⋄]Skolkovo Institute of Science and Technology, Russia
{anwar,biemann,panchenko}@informatik.uni-hamburg.de
[†]Data and Web Science Group, University of Mannheim, Germany
{dmitry,simone}@informatik.uni-mannheim.de
[‡]Samsung R&D Institute Russia
[§]Lomonosov Moscow State University, Russia
narefyev@cs.msu.ru

Abstract

We present our system for semantic frame induction that showed the best performance in Subtask B.1 and finished as the runner-up in Subtask A of the SemEval 2019 Task 2 on unsupervised semantic frame induction (QasemiZadeh et al., 2019). Our approach separates this task into two independent steps: verb clustering using word and their context embeddings and role labeling by combining these embeddings with syntactical features. A simple combination of these steps shows very competitive results and can be extended to process other datasets and languages.

1 Introduction

Recent years have seen a lot of interest in computational models of frame semantics, with the availability of annotated sources like PropBank (Palmer et al., 2005) and FrameNet (Baker et al., 1998). Unfortunately, such annotated resources are very scarce due to their language and domain specificity. Consequently, there has been work that investigated methods for unsupervised frame acquisition and parsing (Lang and Lapata, 2010; Modi et al., 2012; Kallmeyer et al., 2018; Ustalov et al., 2018). Researchers have used different approaches to induce frames, including clustering verb-specific arguments as per their roles (Lang and Lapata, 2010), subject-verb-object triples (Ustalov et al., 2018), syntactic dependency representation using dependency formats like CoNLL (Modi et al., 2012; Titov and Klementiev, 2012), and latent-variable PCFG models (Kallmeyer et al., 2018).

The SemEval 2019 task of semantic frame and role induction consists of three subtasks: (A) learning the frame type of the highlighted verb from the context in which it has been used; (B.1) clustering the highlighted arguments of the verb into specific roles as per the frame type of that verb, e.g., *Buyer*, *Goods*, etc.; (B.2) clustering the arguments into generic roles as per VerbNet classes (Schuler, 2005), without considering the frame type of the verb, i.e., *Agent*, *Theme*, etc.

Our approach to frame induction is similar to the word sense induction approach by Arefyev et al. (2018), which uses tf–idf-weighted context word embeddings for a shared task on word sense induction by Panchenko et al. (2018). In this unsupervised task, our approach for clustering mainly consists of exploring the effectiveness of already available pre-trained models.[1] Main contributions of this paper are:

1. a method that uses contextualized distributional word representations (embeddings) for grouping verbs to frame type clusters (Subtask A);

2. a method that combines word and context embeddings for clustering arguments of verbs to frame slots (Subtasks B.1 and B.2).

The key difference of our approach with respect to prior work by Arefyev et al. (2018) and Kallmeyer et al. (2018) is that we have only used pre-trained embeddings to disambiguate the verb senses and then combined these embeddings with additional features for semantic labeling of the verb roles.[2]

[1]HHMM is an abbreviation for Hansestadt Hamburg, Mannheim, and Moscow. It is chosen to avoid confusion with hidden Markov models.

[2]Our code is available at `https://github.com/uhh-lt/semeval2019-hhmm`.

The remainder of the paper is organized as follows. The methodology and the results for each subtask are discussed in Sections 2, 3 and 4 respectively, followed by the conclusion in Section 5.

2 Subtask A: Grouping Verbs to Frame Type Clusters

In this subtask, each sentence has a highlighted verb, which is usually the predicate. The goal is to label each highlighted verb according to the frame evoked by the sentence. The gold standard for this subtask is based on the FrameNet (Baker et al., 1998) definitions for frames.

2.1 Method

Since sentences evoking the same frame should receive the same labels, we used a verb clustering approach and experimented with a number of pre-trained word and sentence embeddings models, namely Word2Vec (Mikolov et al., 2013), ELMo (Peters et al., 2018), Universal Sentence Embeddings (Conneau et al., 2017), and fastText (Bojanowski et al., 2017). This setup is similar to treating the frame induction task as a word sense disambiguation task (Brown et al., 2011).

We experimented with embedding different lexical units, such as verb (V), its sentence (context, C), subject-verb-object (SVO) triples, and verb arguments. Combination of context and word representations (C+W) from Word2Vec and ELMo turned out to be the best combination in our case.

We used the standard Google News Word2Vec embedding model by Mikolov et al. (2013). Since this model is trained on individual words only and the SemEval dataset contained phrasal verbs, such as *fall back* and *buy out*, we have considered only the first word in the phrase. If this word is not present in the model vocabulary, we fall back to a zero-filled vector. When aggregating a context into a vector, we used the tf–idf-weighted average of the word embeddings for this context as proposed by Arefyev et al. (2018). We tuned these weights on the development dataset.

We used the ELMo contextualized embedding model by Peters et al. (2018) that generates vectors of a whole context. Similarly to fastText (Bojanowski et al., 2017), ELMo can produce character-level word representations to handle out-of-vocabulary words. In all our experiments we used the same pre-trained ELMo model available on TensorFlow Hub.[3] Among all the layers of this model, we used the mean-pooling layer for word and context embeddings.

	Method	Pu F_1	B^3 F_1
🔥	w2v[C+W]norm	**76.68**	**68.10**
⌛	ELMo[C+W]norm	77.03	69.50
⏲	Cluster Per Verb	73.78	65.35
🏆	Winner	78.15	70.70

Table 1: Our results on Subtask A: Grouping Verbs to Frame Type Clusters. Purity F_1-score is denoted as *Pu F_1*, B-Cubed F_1-score is denoted as *B^3 F_1*. 🔥 denotes *our* final submission (# 536426), ⌛ denotes *our* post-competition result, ⏲ denotes a baseline, and 🏆 denotes the submission of the winning team.

2.2 Results and Discussion

We experimented with different clustering algorithms provided by scikit-learn (Pedregosa et al., 2011), namely agglomerative clustering, DBSCAN, and affinity propagation. After the model selection on the development dataset, we have chosen agglomerative clustering for further evaluation. Although both ELMo and Word2Vec showed the best results on the development dataset with single linkage, we opted average linkage after analyzing t-SNE plots (van der Maaten and Hinton, 2008).

Table 1 shows our results obtained on Subtask A. Our final submission (🔥) used agglomerative clustering of normalized vectors obtained by concatenating the context and verb vectors from the Word2Vec model. In particular, we found that the best performance is attained for Manhattan affinity and 150 clusters. During our post-competition experiments (⌛), we found that ELMo performed better than Word2Vec when a higher number of clusters, 235, was specified.

3 Subtask B.1: Clustering Arguments of Verbs to Frame-Specific Slots

In this subtask, each sentence has a set of highlighted nouns or noun phrases corresponding to the slots of the evoked frame. Additionally, each sentence is provided with the same highlighted verb as in Subtask A (Section 2). The goal is to label each highlighted verb according to the evoked frame and to assign each highlighted to-

[3] `https://tfhub.dev/google/elmo/2`

	Method	Pu F_1	B^3 F_1
	Agglomerative Clustering		
🔥	Subtask A: w2v[C+W] Subtask B.2: ID	**62.10**	**49.49**
	Logistic Regression		
⚡	Subtask A: w2v[C+W]norm Subtask B.2: ELMo[C+W+V]+ID+B+123	66.81	55.61
⌛	Subtask A: ELMo[C+W]norm Subtask B.2: w2v[C+W+V]+ID+B+123	68.22	58.61
⏲	Cluster Per Dependency Role	57.99	45.79
🏆	Winner	62.10	49.49

Table 2: Our results on Subtask B.1: Clustering Arguments of Verbs to Frame-Specific Slots. Purity F_1-score is denoted as *Pu F_1*, B-Cubed F_1-score is denoted as *B^3 F_1*. 🔥 denotes *our* final submission (# 535483), ⚡ denotes a supervised *Logistic Regression* submission that does not comply to the task rules, ⌛ denotes *our* post-competition result, ⏲ denotes a baseline, and 🏆 denotes the submission of the winning team.

ken a frame-specific semantic role identifier. The gold standard for this subtask is annotated with FrameNet frames and roles (Baker et al., 1998).

3.1 Method

Since Subtask B.1 asks to assign role labels to highlighted tokens as per the frame type of the verb, we attempted this by merging the output of verb frame types from Subtask A (Section 2) and the output of generic role labels from Subtask B.2 (Section 4). We used `UKN` (unknown) slot identifier for the tokens present in Subtask B.1, but missing in Subtask B.2.

3.2 Results and Discussion

Table 2 shows the results from merging our solutions for Subtasks A and B.2, as described in Sections 2 and 4, correspondingly. For our final submission (🔥), we merged the frame types obtained by clustering the Word2Vec embeddings of the sentence (context, C) and verb (word, W), and the role labels obtained by clustering the vector of inbound dependencies (ID). However, we observed that the logistic regression model demonstrated better performance in Subtask B.2 than any clustering technique we tried, including our final submission (🔥) and the baselines. But this performance was further improved by combining the results from post-competition experiments of Subtask A and Subtask B.2 (⌛).

4 Subtask B.2: Clustering Arguments of Verbs to Generic Roles

In Subtask B.2, similarly to Subtask B.1 (Section 3), each sentence has a set of highlighted nouns or noun phrases that correspond to the slots of the evoked frame. The goal is to label each highlighted token with a high-level generic class, such as *Agent* or *Patient*. However, unlike Subtask B.1, the verb frame labeling part is omitted. The gold standard for this subtask is annotated as according to the VerbNet classes (Schuler, 2005).

4.1 Method

When addressing this subtask, we experimented with combining the embeddings of the word (W) filling the role, its sentence (context, C), and the highlighted verb (V). To handle the out-of-vocabulary roles in the case of Word2Vec embeddings, each role was tokenized and embeddings for each token were averaged. If a token is still not present in the vocabulary, then a zero-filled vector was used as its embedding. During prototyping we developed several features that improved the performance score, namely *inbound dependencies* (ID), which represent the dependency label from the head to the role (dependent) and two trivial baselines: *Boolean* (B) and *123*.

We built a negative one-hot encoding feature vector to represent the *inbound* dependencies of the word corresponding to the role. Thus, for each dependency of the given role (in case of a multiword expression), we fill `-1` if the dependency relationship holds, otherwise `0` is filled. During our experiments for the development test, we also used the outbound dependencies, which represent the dependency label from the role (head) to the dependent words. So we used `-1` for inbound and `1` for outbound. But since they did not perform well in comparison to inbound dependencies, they were not considered for submitted runs.

For the *Boolean* baseline, given the position of the verb in the sentence p_v and the position of the target token p_t, we assign the role `0` to t if $p_v < p_t$, otherwise `1`. For the *123* baseline, we assign its index to each highlighted slot filler. For example, if five slots need to be labelled, the first one will be labelled as `1` and the last one will be labelled as `5`.

4.2 Results and Discussion

Table 3 shows our results on Subtask B.2. We found that the trivial Boolean approach outper-

	Method	**Pu F_1**	**B^3 F_1**
Agglomerative Clustering			
🔥	w2v[C]+ID	**62.00**	**42.10**
⌛	ELMo[C]+ID	50.37	34.89
Logistic Regression			
⚡	ELMo[C+W+V]+ID+B+123	73.14	57.37
⌛	w2v[C+W+V]+ID+B+123	74.36	58.83
	Cluster Per Dependency Role	56.05	39.03
⏲	Boolean Baseline	67.16	46.78
	Inbound Dependencies (ID)	66.05	45.77
🏆	Winner	64.16	45.65

Table 3: Our results on Subtask B.2: Clustering Arguments of Verbs to Generic Roles. Purity F_1-score is denoted as *Pu F_1*, B-Cubed F_1-score is denoted as *B^3 F_1*. 🔥 denotes *our* final submission (# 535480), ⚡ denotes a supervised *Logistic Regression* submission that does not comply to the task rules, ⌛ denotes *our* post-competition result, ⏲ denotes a baseline, and 🏆 denotes the submission of the winning team.

formed LPCFG (Kallmeyer et al., 2018) and all the standard baselines, including cluster per dependency role (`OneClustPerGrType`), on the development dataset.[4]

Similarly to our solution for Subtask A (Section 2), we tried different clustering algorithms to cluster arguments of verbs to generic roles and found that the best clustering performance is shown by agglomerative clustering with Euclidean affinity, Ward's method linkage, and two clusters. Our final submission (🔥) used the combination of inbound dependencies and Word2Vec embedding for sentence (context, C), which performed marginally better than the cluster per dependency role (`OneClustPerGrType`) baseline, but still not better than such trivial baselines as *Boolean* or 123. Replacing Word2Vec with ELMo in our post-competition experiments have lowered the performance further.

In order to estimate our upper bound of the performance, we compared our best-performing clustering algorithm, i.e., agglomerative clustering, to a logistic regression model. We found that the combination of sentence (context, C), target word (W), and verb (V) vectors, enhanced with our other features, shows substantially better results than a simple clustering model (⚡). However, we did not observe a noticeable difference between the performance of the underlying embedding models.

[4]On the development dataset for Subtask B.2, the *Boolean* baseline demonstrated B-Cubed $F_1 = 57.98$, while LPCFG and cluster per dependency role yielded $F_1 = 40.05$ and $F_1 = 50.79$, correspondingly.

As the model was trained on the development dataset that contained 20 roles in contrast to the test set which contained 32 roles, this approach has its limitations due to this difference of the number and meaning of roles. We believe that the performance could be improved using semi-supervised clustering methods, yet during prototyping with the pairwise-constrained k-Means algorithm (Basu et al., 2004) we did not observe any performance improvements.

5 Conclusion

We presented an approach for unsupervised semantic frame and role induction that uses word and context embeddings. It separates the task into two independent steps: verb clustering and role labelling, using combination of these embeddings enhanced with syntactical features. Our approach showed the best performance in Subtask B.1 and also finished as the runner-up in Subtask A of this shared task, and it can be easily extended to process other datasets and languages.

Acknowledgments

We acknowledge the support of the Deutsche Forschungsgemeinschaft (DFG) under the "JOIN-T" and "ACQuA" projects and the German Academic Exchange Service (DAAD). We thank the SemEval organizers for an inspiring shared task and their quick responses to all our questions. We are grateful to four anonymous reviewers who offered useful comments. Finally, we thank the Linguistic Data Consortium (LDC) for the provided Penn Treebank dataset (Marcus et al., 1993).

References

Nikolay Arefyev, Pavel Ermolaev, and Alexander Panchenko. 2018. How much does a word weight? Weighting word embeddings for word sense induction. In *Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue"*, pages 68–84, Moscow, Russia. RSUH.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, pages 86–90, Montréal, QC, Canada. Association for Computational Linguistics.

Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. 2004. Active Semi-Supervision for Pairwise Constrained Clustering. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, SDM 2004, pages 333–344, Lake Buena Vista, FL, USA. SIAM.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Susan Windisch Brown, Dmitriy Dligach, and Martha Palmer. 2011. VerbNet Class Assignment as a WSD Task. In *Proceedings of the Ninth International Conference onComputational Semantics*, IWCS 2011, pages 85–94, Oxford, UK.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Laura Kallmeyer, Behrang QasemiZadeh, and Jackie Chi Kit Cheung. 2018. Coarse Lexical Frame Acquisition at the Syntax–Semantics Interface Using a Latent-Variable PCFG Model. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, *SEM 2018, pages 130–141, New Orleans, LA, USA. Association for Computational Linguistics.

Joel Lang and Mirella Lapata. 2010. Unsupervised Induction of Semantic Roles. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 939–947, Los Angeles, CA, USA. Association for Computational Linguistics.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., Harrahs and Harveys, NV, USA.

Ashutosh Modi, Ivan Titov, and Alexandre Klementiev. 2012. Unsupervised Induction of Frame-Semantic Representations. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 1–7, Montréal, QC, Canada. Association for Computational Linguistics.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

Alexander Panchenko, Anastasia Lopukhina, Dmitry Ustalov, Konstantin Lopukhin, Nikolay Arefyev, Alexey Leontyev, and Natalia Loukachevitch. 2018. RUSSE'2018: A Shared Task on Word Sense Induction for the Russian Language. In *Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue"*, pages 547–564, Moscow, Russia. RSUH.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, NAACL-HLT 2018, pages 2227–2237, New Orleans, LA, USA. Association for Computational Linguistics.

Behrang QasemiZadeh, Miriam R. L. Petruck, Regina Stodden, Laura Kallmeyer, and Marie Candito. 2019. SemEval-2019 Task 2: Unsupervised Lexical Frame Induction. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, Minneapolis, MN, USA. Association for Computational Linguistics.

Karin Kipper Schuler. 2005. *VerbNet: A Broad-coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA.

Ivan Titov and Alexandre Klementiev. 2012. A Bayesian Approach to Unsupervised Semantic Role Induction. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL 2012, pages 12–22, Avignon, France. Association for Computational Linguistics.

Dmitry Ustalov, Alexander Panchenko, Andrei Kutuzov, Chris Biemann, and Simone Paolo Ponzetto. 2018. Unsupervised Semantic Frame Induction using Triclustering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, ACL 2018, pages 55–62, Melbourne, VIC, Australia. Association for Computational Linguistics.

L^{2}F/INESC-ID at SemEval-2019 Task 2: Unsupervised Lexical Semantic Frame Induction using Contextualized Word Representations

Eugénio Ribeiro[1,2], **Vânia Mendonça**[1,2], **Ricardo Ribeiro**[1,3],
David Martins de Matos[1,2], **Alberto Sardinha**[1,2], **Ana Lúcia Santos**[4,5], **Luísa Coheur**[1,2]
[1] INESC-ID Lisboa, Portugal
[2] Instituto Superior Técnico, Universidade de Lisboa, Portugal
[3] Instituto Universitário de Lisboa (ISCTE-IUL), Lisboa, Portugal
[4] Centro de Linguística da Universidade de Lisboa, Portugal
[5] Faculdade de Letras da Universidade de Lisboa, Portugal
`eugenio.ribeiro@l2f.inesc-id.pt, vania.mendonca@tecnico.ulisboa.pt`

Abstract

Building large datasets annotated with semantic information, such as FrameNet, is an expensive process. Consequently, such resources are unavailable for many languages and specific domains. This problem can be alleviated by using unsupervised approaches to induce the frames evoked by a collection of documents. That is the objective of the second task of SemEval 2019, which comprises three subtasks: clustering of verbs that evoke the same frame and clustering of arguments into both frame-specific slots and semantic roles.

We approach all the subtasks by applying a graph clustering algorithm on contextualized embedding representations of the verbs and arguments. Using such representations is appropriate in the context of this task, since they provide cues for word-sense disambiguation. Thus, they can be used to identify different frames evoked by the same words. Using this approach we were able to outperform all of the baselines reported for the task on the test set in terms of Purity F_1, as well as in terms of BCubed F_1 in most cases.

1 Introduction

The Frame Semantics theory of language (Fillmore, 1976) states that one cannot understand the meaning of a word without knowing the context surrounding it. That is, a word may evoke different semantic frames depending on its context. Considering this relation, sets of frame definitions and annotated datasets that map text into the semantic frames it evokes are important resources for multiple Natural Language Processing (NLP) tasks (Shen and Lapata, 2007; Aharon et al., 2010; Das et al., 2014). The most prominent of such resources is the FrameNet (Baker et al., 1998), which provides a set of more than 1,200 generic semantic frames, as well as over 200,000 annotated sentences in English. However, this kind of resource is expensive and time-consuming to build, since both the definition of the frames and the annotation of sentences require expertise in the underlying knowledge. Furthermore, it is difficult to decide both the granularity and the domains to consider while defining the frames. Consequently, such resources only exist for a reduced amount of languages (Boas, 2009) and even English lacks domain-specific resources in multiple domains.

The problem of building semantic frame resources can be alleviated by using unsupervised approaches to induce the frames evoked by a collection of documents. The second task of SemEval 2019 aims at comparing unsupervised frame induction systems for building semantic frame resources for verbs and their arguments (Qasemi Zadeh et al., 2019). It is split into three subtasks. The first, Task A, focuses on clustering instances of verbs according to the semantic frame they evoke while the others focus on clustering the arguments of those verbs, both according to the frame-specific slots they fill, on Task B.1, and their semantic role, on Task B.2.

In this paper, we address the three subtasks by following an approach that takes advantage of the recent developments on the generation of contextualized word representations (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2018). Such representations are able to disambiguate different word senses by varying the position of a word in the embedding space according to its context. This ability is important in the context of semantic frame induction, since different word-senses typically evoke different frames. To identify words that evoke the same frame or have the same role, our approach consists of clustering their representations by applying the Chinese Whispers algorithm (Biemann, 2006) to a similarity-based graph. This way, we do not need to define the number of clusters and there is no bias towards the generation of clusters of similar size.

In the remainder of the paper, we start by providing an overview of previous studies related to the task, in Section 2. Then, in Section 3, we describe our approach and explain how it differs from previous approaches. Section 4 describes our experimental setup. The results of our experiments are presented and discussed in Section 5. Finally, Section 6 summarizes the conclusions of our work and provides pointers for future work.

2 Related Work

Following the motivation described in the previous section, previous studies have employed unsupervised approaches for the induction of semantic frames and roles. However, most studies have focused on semantic role induction. For instance, Titov and Klementiev (2012) proposed two models based on the Chinese Restaurant Process (Ferguson, 1973). The factored model induces semantic roles for each predicate independently using an iterative clustering approach, starting with one cluster per argument. On the other hand, the coupled model takes into consideration a distance-dependent prior shared among different predicates. Arguments from different predicates are then used as vertices of a similarity graph and each argument selects another argument as a member of the same cluster based on that similarity. Overall, the coupled model performs slightly better than the factored one. In both cases, each argument is represented by a set of syntactic features – sentence voice, argument position, syntactic relation, and existing prepositions.

Lang and Lapata (2014) proposed a graph partitioning approach over a multilayer graph. Each layer corresponds to a feature, i.e., each pair of vertices (arguments) is connected through multiple edges, each corresponding to their similarity according to that feature. Then, two clustering approaches were considered, achieving similar results. The first is an adaptation of agglomerative clustering to the multilayer setting. Instead of combining the similarity values into a single score, it clusters the arguments in each layer and then combines the obtained scores into a multilayer score. Clusters with greater multilayer similarity are then merged together, with larger clusters being prioritized. The second clustering approach consists of propagating cluster membership along the graph edges. In both cases, the combination of the scores of each layer is based on a set of conditions, in order to avoid having to learn or guess weights for each feature.

In contrast to the previous approaches, Titov and Khoddam (2015) proposed a reconstruction-error maximization framework which comprises two main components: an auto-encoder, responsible for labeling arguments with induced roles, and a reconstruction model, which takes the induced roles and predicts the argument that fills each role, i.e., it tries to reconstruct the input. The learning error is obtained by comparing the reconstructed argument to the original one. This enables the use of a larger feature set and more complex features, similarly to supervised approaches.

Concerning frame induction, Ustalov et al. (2018a) proposed a graph-based approach for the triclustering of Subject-Verb-Object (SVO) triples extracted using a dependency parser. Each vertex in the graph is the SVO triple, represented by the concatenation of word embeddings for the three elements. Vertices are connected to their k-nearest neighbours (k=10) according to their cosine similarity. The clusters are then generated using the Watset fuzzy graph clustering algorithm (Ustalov et al., 2017), which induces word-sense information in the graph before clustering. For each cluster, the corresponding triframe is generated by aggregating the subjects, verbs, and objects into separate sets and generating a triple using those sets. This approach outperformed hard clustering approaches, as well as topic-based approaches, such as LDA-Frames (Materna, 2012).

3 Induction Approach

Considering the subtasks we are approaching, we must use an approach that is able to induce not only semantic roles, but also semantic frames and its slots. In this sense, of the approaches described in the previous section, the triclustering approach proposed by Ustalov et al. (2018a) is the only one able to induce frames. However, in the context of our task, it has two major flaws. First, it focuses on the clustering of SVO triples, i.e., a frame is defined by a head and two slots. In our case, each instance has a variable number of arguments. Thus, the triclustering approach is not appropriate. Furthermore, since the arguments are clustered in combination with the verb, this approach is particularly inappropriate for semantic role induction. The second flaw is related to the approach used for inducing word-sense information, which requires a thesaurus to provide synonymity information. Such resources must be manually built and, thus, may not be available for every language or lack domain-specific information.

We approach the first flaw by clustering the verb and its arguments independently. This way, we are able to cluster the instances of verbs to identify the frame heads, as required for Task A, and the instances of arguments to identify semantic roles, as required for Task B.2. To identify the slots of each frame, as required for Task B.1, we combine the clusters of the verbs with those of the arguments.

To deal with the second flaw, we replace the per-word embeddings used by Ustalov et al. (2018a) with contextualized word representations. These include information concerning the context in which a word appears and, thus, the position of a word in the embedding space varies according to that context. By using such representations, we are able to discard the fuzzy clustering approach used by Ustalov et al. (2018a) to induce word-sense, since it is revealed by the contextual variations of the representation of a word. Therefore, a hard clustering algorithm can be applied directly.

Algorithm 1 Induction Approach

```
Input: $T$ // The set of head tokens to cluster
Input: EMBED // The contextualized embedding approach
Input: THRESH // The function for computing the neighboring threshold
Output: $C$ // The set of clusters
1: $V \leftarrow \{\text{EMBED}(t) : t \in T\}$ // The whole sentence is required for embedding generation
2: $D \leftarrow \{1 - \cos(\theta_{v,v'}) : (v, v') \in V^2, v \neq v'\}$ // $\theta_{v,v'}$ is the angle between the two vectors
3: $t \leftarrow \text{THRESH}(D)$
4: $E \leftarrow \{(v, v', D_{v,v'}) : (v, v') \in V^2, v \neq v', D_{v,v'} < t\}$ // The edge is weighted with the cosine distance between the vertices
5: $C \leftarrow \text{CHINESEWHISPERS}(V, E)$
6: return $C$
```

Our approach is summarized in Algorithm 1. It starts by generating the contextualized representation of each instance to be clustered. In cases where the verb or argument to cluster consists of multiple words, we use a dependency parser to identify the head word and use its contextualized representation, since it contains information from the other words. Then, in order to build a graph, we compute the pairwise distances between the instances. These distances are used to decide which instances are considered neighbors. Since each instance is represented as a vector in the embedding space, we use the cosine distance. Moreover, since using a fixed number of neighbors is not realistic, we decided to use a threshold based on this distance. This threshold defines the granularity of the clusters and varies according to the set of instances. Instead of using a fixed threshold, we define it based on the parameters of the pairwise distances distribution. The actual combination of the parameters varies according to the subtask and is further discussed in the subsections below. Finally, to obtain the clusters, we apply the Chinese Whispers algorithm (Biemann, 2006) on a graph where the vertices are the instances and the edges connect neighbor instances. The weight of each edge is given by the distance between neighbors. We use the Chinese Whispers algorithm since it chooses the number of clusters on its own and is able to handle clusters of different sizes, thus being appropriate for the task. Furthermore, it has been proved successful in NLP clustering tasks.

3.1 Verb Clustering

The first subtask focuses on clustering verbs that evoke the same frame. The number of frames evoked in a set of documents is typically larger than the number of semantic roles and even larger in comparison to the number of slots per frame. Thus, a lower neighboring threshold is required to achieve such granularity. In our experiments, we achieved the best results when defining the neighboring threshold for clustering verbs, t_f, as

$$t_f = \frac{\mu + \sigma}{2}, \tag{1}$$

where μ and σ are the mean and standard deviation of the pairwise distance distribution, respectively. Using this threshold may lead to the induction of frames with different granularity, depending on the sense similarity between the verbs present in the dataset. However, if the induced frames are considered too abstract, the approach can be applied hierarchically on the instances of each cluster to obtain finer-grained frames.

3.2 Argument Clustering

Both the second and third subtasks focus on clustering arguments. However, while the second focuses on doing so in a per-frame manner to induce its slots (frame elements), the third focuses on clustering them independently of the frame, i.e., to induce generic semantic roles. In the first case it would make sense to cluster the arguments of verbs that evoke each frame independently of the others. However, that may not be feasible on small datasets. Thus, we opted for clustering all the arguments together in both cases. The slot clusters for the second subtask are then given by the combination of the verb and argument clusters. Thus, this approach considers that slots are per-frame specializations of the semantic roles, which is accurate in most situations.

As previously stated, the number of semantic roles is typically smaller than the number of frames. Thus, a higher neighboring threshold can be used. In our experiments, we achieved the best results when defining the neighboring threshold while clustering arguments, t_a, as

$$t_a = \mu - 1.5\sigma. \qquad (2)$$

Finally, since the arguments are highly dependent on the verb, we also performed experiments in which we combined the contextualized representation of the argument with that of the verb before applying the clustering approach.

4 Experimental Setup

In this section we describe our experimental setup in terms of data, implementation details, and evaluation metrics and baselines.

4.1 Dataset

In our experiments, we used the dataset provided by the task organization, built with sentences from the Penn Treebank 3.0 (Marcus et al., 1993), and annotated with FrameNet frames (Task A), frame elements or slots (Task B.1) and generic semantic roles (Task B.2). The development set consists of 600 verb-argument instances, 588 sentences and 1,211 arguments. The (blind) test set comprises 4,620 verb-argument instances, 3,346 sentences, 9,466 arguments labeled for semantic role and 9,510 arguments labeled for frame slot. Additionally, morphosyntactic information is provided in the CoNLL-U format (Buchholz and Marsi, 2006).

4.2 Implementation Details[1]

In our experiments we compared the performance of two approaches to generate the contextualized word representations. The first, ELMo (Peters et al., 2018), is based on bi-directional LSTMs (Hochreiter and Schmidhuber, 1997) and was the first approach to generate contextualized representations. Its output provides a context-free representation of the word and context information at two levels. In our experiments we use the sum of all information, since it leads to variations of the context-free representation according to the context. The second representation, BERT (Devlin et al., 2018), is based on the Transformer architecture (Vaswani et al., 2017) and currently leads to state-of-the-art results on multiple benchmark NLP tasks. Its output can be extracted from a single layer or the multiple layers included in the model. Contrarily to the ELMo layers, these do not have an associated semantics. Thus, we use the output of the last layer, since it contains information from all that precede it. In both cases we used pre-trained models. To obtain embedding vectors with the same dimensionality, 1,024, we used the ELMo model provided by the AllenNLP package (Gardner et al., 2017) and the large uncased BERT model provided by its authors.

To apply the Chinese Whispers algorithm, we relied on the implementation by Ustalov et al. (2018b), which requires the graph to be built using the NetworkX package (Hagberg et al., 2004). We did not use weight regularization and performed a maximum of 20 iterations. Furthermore, in order to avoid result changes based on non-deterministic factors, we fixed the random seed as 1337.

Finally, to obtain the syntactic dependencies used to determine the head token of multi-word verbs or arguments, we used the annotations provided with the task dataset.

4.3 Baselines

For comparison purposes, in addition to our results, we report the baselines provided by the task scorer. For the frame induction subtask (Task A), the baseline consists of assigning each verb lemma to a frame (*Lemma*). For the semantic role induction subtask (Task B.2), arguments are assigned to clusters according to their syntactic relation to the head verb (*Dep*). For the frame slot induction subtask (Task B.1), the previous baselines are combined by assigning each pair of verb lemma and argument's syntactic dependency to a cluster (*Lemma + Dep*). On the test set, we also consider a random assignment to the gold number of clus-

[1] https://gitlab.l2f.inesc-id.pt/eugenio/find/

	Approach	#C	Purity	inv-Purity	Purity F_1	B^3 Precision	B^3 Recall	B^3 F_1
Task A	**ELMo**	32	93.17	96.50	**94.80**	89.06	95.63	**92.23**
	BERT	72	89.67	84.00	86.74	83.17	77.77	80.38
	BL: Lemma	35	93.50	85.67	89.41	90.22	79.63	84.60
Task B.1	**ELMo**	72	68.35	72.40	70.31	57.60	64.18	60.72
	BERT	170	52.98	72.98	61.39	46.82	62.77	53.64
	ELMo + Verb	72	68.35	72.40	70.31	57.60	64.18	60.72
	BL: Lemma + Dep	136	84.30	70.74	**76.93**	78.71	58.36	**67.03**
Task B.2	**ELMo**	11	62.23	69.01	65.44	46.75	56.33	51.10
	BERT	72	48.35	83.97	61.36	38.94	72.86	50.75
	ELMo + Verb	140	70.17	43.80	53.93	62.20	23.27	33.87
	Dep + PoS	66	65.95	29.26	40.53	55.61	20.05	29.47
	BL: Dep	22	67.93	71.32	**69.59**	53.31	57.67	**55.41**

Table 1: Results obtained on the development set. The baselines are identified with *BL*.

ters as a baseline. Due to space constraints, we do not report the results of the remaining baselines proposed by Kallmeyer et al. (2018).

We report the results of an additional baseline for Task B.2 which considers both the argument's syntactic relation to the head verb and its Part-of-Speech (POS) tag (*Dep + POS*).

4.4 Evaluation metrics

We report our results using the metrics defined for the task: number of clusters (*#C*), purity, inverse-purity, and their harmonic mean (Purity F_1), as proposed by Steinbach et al. (2000), and BCubed (B^3) precision, recall, and F_1, as proposed by Bagga and Baldwin (1998).

5 Results

The results obtained on the development set are reported in Table 1. We can see that using ELMo to obtain the contextualized word representations leads to better results than BERT on every subtask. This is somewhat surprising since BERT is the state-of-the-art approach to generate contextualized representations. A possible explanation may lie in the fact that the two levels of ELMo which provide context information can be related to syntax and semantics (Peters et al., 2018), making them highly related to the task. On the other hand, the information provided by BERT representations is not as easy to categorize. Moreover, in every case, the number of clusters is underestimated when using ELMo and overestimated when using BERT.

On the frame induction subtask (Task A), our approach surpasses every baseline, but only when using ELMo embeddings. The lemma baseline is surpassed by over 5 percentage points on Purity F_1 and 7.5 on BCubed F_1. The same is not true on the other tasks, with the clustering based on the dependency relation between the argument and verb achieving the best results. It outperforms our approach in terms of both F_1 metrics by around 6.5 percentage points on the slot induction subtask (Task B.1) and around 4 points on the semantic role induction subtask (Task B.2). We believe that this happens because the development set is small and the kind of arguments does not vary much.

Combining the verb representation with that of the argument leads to worse results on Task B.2, since it is clustering the semantic roles per verb. On Task B.1, the result is the same as without using the verb representation, which suggests that the information provided by the verb is not able to improve the induced slots, but only to attribute them to the corresponding frame.

The approach which combines the dependency relation with the POS tag obtains worse results on Task B.2, as it leads to additional partitioning of the clusters. Thus, a large number of clusters is generated, which is not consistent with the nature of semantic roles.

The results obtained on the test set are reported in Table 2. We only submitted the clusters obtained using ELMo, since it outperformed BERT on the development set. Similarly, we did not consider the combination of verb and argument representation for the argument clustering tasks. However, we assessed the performance of the baseline based on the dependency relation and the POS tag.

On Task A, our approach surpasses all the baselines in terms of Purity F_1, but by less than 2 percentage points. In fact, it has a similar perfor-

	Approach	#C	Purity	inv-Purity	Purity F_1	B^3 Precision	B^3 Recall	B^3 F_1
Task A	**ELMo**	222	72.84	77.84	**75.25**	61.25	69.96	65.32
	BL: Lemma	273	82.16	66.95	73.78	75.98	57.33	**65.35**
	BL: Random	**149**	15.30	5.74	8.34	6.82	3.85	4.92
Task B.1	**ELMo**	526	58.26	64.30	**61.13**	44.79	53.21	**48.64**
	BL: Lemma + Dep	1203	78.46	45.99	57.99	71.11	33.77	45.79
	BL: Random	**436**	11.25	6.09	7.90	6.07	4.82	5.37
Task B.2	**ELMo**	6	58.29	71.19	**64.10**	36.80	60.15	**45.66**
	Dep + PoS	159	57.39	26.25	36.03	41.41	15.07	22.1
	BL: Dep	37	61.44	51.53	56.05	40.89	37.33	39.03
	BL: Random	**32**	34.77	4.85	8.51	21.92	3.46	5.98

Table 2: Results obtained on the test set. The baselines are identified with *BL*.

mance to the lemma baseline in terms of BCubed F_1. This happens because it overestimates the number of clusters, which suggests that the problem may be related to the threshold. However, using a threshold that leads to the induction of a number of frames similar to the gold standard ends up generating clusters of lower quality. This suggests that additional features must be introduced.

On the remaining tasks, our approach performs better than every baseline, which supports the claim that the better performance of the clustering approach based on the dependency relation on the development set is due to the limited variation in the kinds of argument present in that set. We observed an improvement of around 4 percentage points on Task B.1 on both F_1 metrics, and above 8 percentage points on Purity F_1 and nearly 7 on BCubed F_1 on Task B.2.

Once again, the approach which combines the dependency relation with the POS tag leads to worse results on Task B.2, due to additional partitioning of the clusters. In this case, the number of semantic roles is even more overestimated.

6 Conclusions

In this paper we presented our approach on unsupervised semantic frame, slot, and role induction in the context of the second task of SemEval 2019. The approach is based on the clustering of contextualized word representations of verbs and arguments. Using such representations is appropriate for the task since they provide word-sense information which is important for distinguishing the evoked frames.

We were able to achieve results that surpassed or performed on par with every baseline proposed for the three subtasks on the test set. However, the results are far from perfect and below those achieved by more complex approaches on the task, which suggests that the contextualized representations on their own are not able to provide all the information required to perform an accurate frame induction. Thus, as future work, we intend to assess the cases that our approach fails to cluster, and introduce additional features that provide relevant information for those cases, either by using a weighted combination of per-feature distance functions or a multilayer graph similar to that proposed by Lang and Lapata (2014).

Furthermore, since the number of instances in the test set is larger than in the development set, it may be feasible to apply a per-frame clustering approach for the slot induction task. This way, the induced slots are no longer mere specifications of the generic semantic roles.

Finally, although the number of semantic roles is not consensual in the literature, there is a set of core semantic roles which is common to every theory. Thus, it would be interesting to take advantage of that information to apply clustering approaches with a pre-defined number of clusters for semantic role induction. In fact, it would be interesting to explore other clustering approaches on every task and compare their performance with that of the Chinese Whispers algorithm.

Acknowledgements

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2019. Vânia Mendonça is funded by an FCT grant with reference SFRH/BD/121443/2016. The use of the corpus was licensed by the Linguistic Data Consortium (LDC).

References

Roni Ben Aharon, Idan Szpektor, and Ido Dagan. 2010. Generating Entailment Rules from Framenet. In *ACL*, volume 2, pages 241–246.

Amit Bagga and Breck Baldwin. 1998. Algorithms for Scoring Coreference Chains. In *LREC*, pages 563–566.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *ACL/COLING*, volume 1, pages 86–90.

Chris Biemann. 2006. Chinese Whispers: An Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems. In *Workshop on Graph-based Methods for Natural Language Processing*, pages 73–80.

Hans C. Boas, editor. 2009. *Multilingual FrameNets in Computational Lexicography: Methods and Applications*. Mouton de Gruyter.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *CoNLL*, pages 149–164.

Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-Semantic Parsing. *Computational Linguistics*, 40(1):9–56.

Jacob Devlin, Ming-wei Chang, Lee Kenton, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, abs/1810.04805.

Thomas S. Ferguson. 1973. A Bayesian Analysis of Some Nonparametric Problems. *The Annals of Statistics*, 1(2):209–230.

Charles J. Fillmore. 1976. Frame Semantics and the Nature of Language. *Annals of the New York Academy of Sciences*, 280(Origins and Evolution of Language and Speech):20–32.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. AllenNLP: A Deep Semantic Natural Language Processing Platform. *CoRR*, abs/1803.07640.

Aric Hagberg, Dan Schult, and Pieter Swart. 2004. NetworkX. GitHub.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Laura Kallmeyer, Behrang Qasemi Zadeh, and Jackie Chi Kit Cheung. 2018. Coarse Lexical Frame Acquisition at the Syntax–Semantics Interface Using a Latent-Variable PCFG Model. In *SEM*, pages 130–141.

Joel Lang and Mirella Lapata. 2014. Similarity-Driven Semantic Role Induction via Graph Partitioning. *Computational Linguistics*, 40(3):633–670.

Mitchell Marcus, Beatrice Santorini, and Mary Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):330–331.

Jiří Materna. 2012. LDA-Frames: An Unsupervised Approach to Generating Semantic Frames. In *CICLing*, pages 376–387.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *NAACL-HLT*, volume 1, pages 2227–2237.

Behrang Qasemi Zadeh, Miriam R L Petruck, Stodden Regina, Laura Kallmeyer, and Marie Candito. 2019. Semeval 2019 task 2: Unsupervised lexical frame induction. In *SemEval@NAACL-HLT*. The Association for Computer Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training. Preprint.

Dan Shen and Mirella Lapata. 2007. Using Semantic Roles to Improve Question Answering. In *EMNLP-CoNLL*, pages 12–21.

Michael Steinbach, George Karypis, and Vipin Kumar. 2000. A Comparison of Document Clustering Techniques. In *KDD Workshop on Text Mining*.

Ivan Titov and Ehsan Khoddam. 2015. Unsupervised Induction of Semantic Roles within a Reconstruction-Error Minimization Framework. In *NAACL-HLT*, volume 1, pages 1–10.

Ivan Titov and Alexandre Klementiev. 2012. A Bayesian Approach to Unsupervised Semantic Role Induction. In *EACL*, volume 1, pages 12–22.

Dmitry Ustalov, Alexander Panchenko, and Chris Biemann. 2017. Watset: Automatic Induction of Synsets from a Graph of Synonyms. In *ACL*, volume 1, pages 1579–1590.

Dmitry Ustalov, Alexander Panchenko, Andrei Kutuzov, Chris Biemann, and Simone Paolo Ponzetto. 2018a. Unsupervised Semantic Frame Induction using Triclustering. In *ACL*, volume 2, pages 55–62.

Dmitry Ustalov et al. 2018b. Chinese Whispers for Python. GitHub.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *NIPS*, pages 5998–6008.

BrainEE at SemEval-2019 Task 3: Ensembling Linear Classifiers for Emotion Prediction

Vachagan Gratian
Universität Stuttgart
vgratian@utopianlab.am

Abstract

We present a homogeneous ensemble of linear perceptrons trained for emotion classification as part of the SemEval-2019 shared-task 3. The model uses a matrix of probabilities to weight the activations of the base-classifiers and makes a final prediction using the *sum rule*. The base-classifiers are multi-class perceptrons utilizing character and word n-grams, part-of-speech tags and sentiment polarity scores. The results of our experiments indicate that the ensemble outperforms the base-classifiers, but only marginally. In the best scenario our model attains an F-Micro score[1] of 0.672, whereas the base-classifiers attained scores ranging from 0.636 to 0.666.

1 Introduction

Our task is to detect emotions in multi-turn chat messages (see examples in table 1). The four emotion categories the model has choose from are `happy, sad, angry` and `others`. A major caveat of the task is the imbalance of class distribution in the dataset, as described in 4.1. The dataset, as well as the task itself are described in detail in (Chatterjee et al., 2019).

We choose to deploy ensemble of linear classifiers for this task, rather than a single model for a number of reasons. Firstly, given the inherent ambiguity of emotions (Brainerd, 2018) we expect that ensembles are better suited for any emotion prediction task. Secondly, it has been shown that ensembles are more immune to overfitting in similar tasks (Dong and Han, 2004). And finally, a single model trained on a large number of feature sets, tend to perform significantly worse than an ensemble where each model is trained on a different subset (or combinations) of feature types.

[1]This is a custom F-Micro score. See more details under 4.3 Evaluation

Conversation	Emotion
A: Yes B: How so? A: Don't message me ever	angry
A: I am fine B: I am good how is ur week A: I am single	others

Table 1: Two samples from the dataset with *angry* and *others* respectively as gold labels.

For this purpose, we deploy *BrainT*, a multi-class perceptron model utilizing word n-grams and POS-tags, built and trained for implicit emotion detection in Tweets (Gratian and Haid, 2018). In the current scenario, we extend the feature sets of *BrainT* with character n-grams and Sentiment polarity scores. We combine $n = 11$ and $n = 5$ classifiers into an ensemble model where a final prediction is made based on the activations. Our model also calculates a matrix of probabilities used to weigh the input activations. Each element in the matrix is the probability of a given node making correct prediction for a given emotion class. In the initial experiments the nodes are trained on the full train data. In the second group of experiments, nodes are assigned a random subsets of the train data separately. We hope that this will promote diversity in the base-classifiers and boost the performance of the ensemble.

The results of our experiments indicate that in both cases the ensemble outperforms the base-classifiers, however only slightly. In the following sections we describe the architecture of the model, the actual results on the SemEval shared-task. Finally we suggest ways to maximize the effectiveness of ensemble models as ideas for future work.

2 Related Work

Ensemble learning aims at exploiting the ”shared knowledge” of multiple classifiers based on Statistical Learning theory. A theoretical analysis of ensemble learning using linear perceptrons, can be found in (Hara and Okada, 2005) and (Miyoshi et al., 2005). The authors demonstrate that the generalization error of ensemble learning depends on (hence, can be calculated from) two cosine measures: the similarity between the base-classifiers and the training data and the mutual similarity of the base-classifiers. In plain English, to maximize the performance of the ensemble model, we want to increase the accuracy of the base-classifiers but in such a way that we promote diversity in the base-classifiers.

A simple way of combining the base-classifiers is to take the average of their weights after training. A more common approach is to exploit the output activations using different techniques. In (Xia et al., 2011a) three such techniques are analyzed for sentiment classification: fixed combination, weighted combination and meta-classifier combination. The authors found fixed combination to be the weakest of all three, while weighted combination and meta-classifier added on average 3-4% improvement over the performance of the best base-classifier.

In our work, we deploy the weighted combination technique with the addition of a learned probability matrix, as described below.

3 The Model

3.1 Base-Classifiers

We use as base-classifier the linear perceptron model described in (Gratian and Haid, 2018) which reduces the task of multi-class prediction into $|Y|$ binary classification problems (where $|Y|$ is the number of emotion classes) following the ”one-against-all” approach described by Xia et al. (2011b).

The output of each base-classifier is a vector α of size $|Y|$ corresponding to the number of emotions. Before passing this vector to the ensemble model, each activation α_y is calibrated as follows:

$$\hat{\alpha}_y = \frac{\alpha_y}{\sum\limits_{\hat{y} \in Y} \alpha_{\hat{y}}}$$

By doing so, each $\hat{\alpha}_y$ can be treated as a confidence level of the node that the instance x_i expresses the emotion class y. The advantage of this approach is that even when the true emotion class is not the one predicted by the node (i.e., it is not the highest activation), it can still contribute to the true class being predicted by the ensemble if it has a positive value.

3.2 Ensemble

The ensemble model M represents a matrix of probabilities of size $n \times m$:

$$M = \begin{pmatrix} \varphi_{1,1} & \varphi_{1,2} & \cdots & \varphi_{1,m} \\ \varphi_{2,1} & \varphi_{2,2} & \cdots & \varphi_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_{n,1} & \varphi_{n,2} & \cdots & \varphi_{n,m} \end{pmatrix}$$

The ij-th element of this matrix is the probability that i-th node’s prediction for j-th class is correct. This parameter is initialized to 1 and is learned during training as:

$$\varphi_{i,j} = \frac{|R_{ij}|}{|R_{ij}| + |R'_{ij}|}$$

where $|R_{ij}|$ and $|R'_{ij}|$ are respectively the correct and incorrect predictions of node i for class j during training. This probability value, thus, corresponds to the Precision metric.

The input of the model is a matrix of equal size: those are the activations of n nodes, each a vector of size m (or $|Y|$). We weigh these activations by the probabilities learned by the model by taking the Hadamard product of the two matrices. The final prediction is made by the ensemble following the *sum rule* as described by (Xia et al., 2011b). The predicted class $\hat{y}$ is the one that has the highest sum of weighted activations:

$$\hat{y} = \text{argmax}_{i=1}^{m} \sum_{j=1}^{n} \varphi_{i,j}\, \hat{a}_{i,j}$$

4 Experiments

4.1 Dataset

The dataset we use is provided by the SemEval2019 shared-task 3 and is described in detail by (Chatterjee et al., 2019). It contains a train set with 30,160 and a test set with 5,509 conversations. In both sets the emotion class *others* is disproportionately overrepresented. Moreover, there

Angry	Sad	Happy	Others	Total
5,506	5,463	4,243	14,948	30,160
18.3%	18.1%	14.1%	49.6%	100%

Table 2: Class distribution in the train dataset.

Angry	Sad	Happy	Others	Total
298	250	284	4,677	5,509
5.4%	4.5%	5.2%	84.9%	100%

Table 3: Class distribution in the test dataset.

is a significant difference between the class distributions in the train and test datasets as can be seen in tables 2 and 3 imposing an additional challenge for the classification task.

The evaluation metric of the shared task is a custom F-micro measure which takes only into account the three emotion classes (`happy`, `angry`, `sad`) and disregards the overrepresented class `others`.

4.2 Preprocessing

As in the previous experimental setup, we apply minimal preprocessing. We don't normalize tokens and don't filter stopwords as this proved to decrease system performance in our previous experiments. We treat the 3 turns in each conversation as one stream of tokens by concatenating them using the special token $\langle STOP \rangle$.

4.3 Features

Our feature types are word and character n-grams, as well as POS tags extracted with the NLTK part-of-speech tagger and polarity scores from the `Sentiment Classification using WSD` library [2].

The word n-grams include unigrams, bigrams, trigrams and tetragrams where one token is replaced with the placeholder $\langle SKIP \rangle$ tag as this feature type proved to be highly efficient in our previous experiments.

The list of feature types used in our experiments is in table 4.

4.4 Experimental Setup

We assign each node 2 to 4 feature types. In the preparatory stage of the experiments we train nodes with different combinations of the feature types and select the 11 highest ranking nodes. Table 6 lists these nodes.

[2]The library is free-software and is available online: `https://github.com/kevincobain2000/sentiment_classifier`

Feature Set	Description
1GR 2GR 3GR 4GR-S1	word n-grams
1CH 2CH 3CH	character n-grams
POS SENTA	Lexicon-based

Table 4: The feature types utilized by the base-classifiers.

	False Positives	False Negatives
Happy, Sad, Angry	0.8	1.5
Others	0.1	0.5

Table 5: Learning rates

To overcome overrepresantation of the class `others` we apply a lower learning rate for this class. We furthermore apply a higher learning rate for false negatives than false positives, since in the preparatory experiments all nodes showed a significantly lower Recall than Precision. Table 5 lists those learning rates.

Finally, we test the ensemble model in two experimental setups: *uniform learning* and *distributed learning*. In the first scenario, the entire train data is used to train the 11 nodes. Either all 11 node activations are passed to the ensemble or only those of the 5 highest performing nodes. In the second scenario, each node is assigned and trained on a random 50% subset of the train data.

For all our experiments we choose the number of epochs to be 60.

5 Results

5.1 Uniform Learning

In all of our experiments the ensemble performs only slightly better than the best performing node(s). The results of the experiment with uniform training are in Table 6. We observe that reducing the number of nodes from 11 to 5, decreases Precision of the ensemble, but increases Recall, however in both cases the difference is insignificant.

Node	Precision	Recall	F-micro
1GR_3GR_3CH_POS	0.604	0.672	0.636
2GR_4GR-S1_3CH_POS	0.627	0.675	0.65
2GR_4GR-S1_3CH	0.622	0.69	0.654
1GR_2GR	0.613	0.706	0.656
2GR_SENTA	0.661	0.661	0.661
2GR_3CH_SENTA	0.649	0.677	0.662
1GR_2GR_POS	0.632	0.695	0.662
1GR_2GR_SENTA	0.636	0.692	0.663
1GR_2GR_3CH_SENTA	0.632	0.697	0.663
1GR_2GR_1CH	0.638	0.696	0.666
1GR_2GR_3CH	0.631	0.704	0.666
ENSEMBLE_N=5	**0.640**	**0.700**	**0.672**
ENSEMBLE_N=11	**0.649**	**0.694**	**0.671**

Table 6: Results for Exp 1 with 11 nodes and uniform training.

Node	Precision	Recall	F-micro
1GR_3GR_3CH_POS	0.594	0.614	0.604
2GR_SENTA	0.635	0.579	0.606
2GR_4GR-S1_3CH_POS	0.6	0.629	0.614
2GR_4GR-S1_3CH	0.589	0.649	0.617
2GR_3CH_SENTA	0.639	0.626	0.633
1GR_2GR_3CH	0.607	0.665	0.635
1GR_2GR_1CH	0.615	0.669	0.641
1GR_2GR_SENTA	0.622	0.666	0.643
1GR_2GR_3CH_SENTA	0.623	0.663	0.643
1GR_2GR	0.616	0.675	0.644
1GR_2GR_POS	0.623	0.668	0.645
ENSEMBLE_N=11	**0.648**	**0.666**	**0.657**

Table 7: Results for Exp 2 with 11 nodes and distributed training.

We also observe that while the ensemble outperforms the nodes in the F-micro measure, it has a lower Precision and Recall than at least one of the nodes.

5.2 Distributed Learning

In the second experiment each node is trained on a 50% random subset of the train data. We observe a drop in the performance of both the ensemble and the nodes, although the ensemble outperforms the nodes with a slightly larger margin.

Compared to the results of uniform learning, we see that the ensemble has roughly the same Precision, but a lower Recall. However when we compare the performance of the ensemble with the base-classifiers, we see that the ensemble now has now a higher Precision score than any of the nodes. This indicates that the ensemble benefits more from the "shared knowledge" of the base-classifiers.

6 Discussion

The goal of our experiments was to build an ensemble that makes better predictions than any of the base-classifiers individually. While the results of our experiments prove this to be a success, they also indicate that the ensemble exploits the strengths of the nodes weakly. For most of the emotion classes, the ensemble underperforms at least one of the nodes.

This disparity is especially vivid in the Recall measure. We presume that this due to the fact that the probabilities matrix learned by the model reflects only Precision, not Recall. As a future improvement to the model, we could adapt the probabilities to reflect Recall as well.

7 Future Work

In our future work we want to adopt a different approach to ensemble learning. Firstly, we think an important starting point should be a concrete estimation of the ensemble's upper bound performance given n base-classifiers. This can then serve as banchmark to evaluate the actual performance of an ensemble model. In most, if not all, real-world situations, the probability that a node n_i makes a correct prediction for an instance x_j will always be conditional to the probability of another node n_j. This means that the upper boundary of the ensemble model depends on the conditional probabilities of its nodes.

This implies that in our future work we will describe ensemble learning as the task to minimize joint entropy of the base-classifiers in addition to maximizing accuracy.

8 Conclusion

In this paper we describe an ensemble model trained for emotion classification. We evaluate our model on uniform and distributed learning of the train data. The results of the experiments indicate that while the model outperforms the strongest model, it benefits weakly from the strengths and variance of the base-classifiers.

References

C. J. Brainerd. 2018. The emotional-ambiguity hypothesis: A large-scale test. *Psychological Science*, 29(10):1706–1715. PMID: 30130163.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Yan-Shi Dong and Ke-Song Han. 2004. A comparison of several ensemble methods for text categorization. pages 419– 422.

Vachagan Gratian and Marina Haid. 2018. Braint at iest 2018: Fine-tuning multiclass perceptron for implicit emotion classification. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 243–247. Association for Computational Linguistics.

Kazuyuki Hara and Masato Okada. 2005. Ensemble learning of linear perceptrons: On-line learning theory. *Journal of The Physical Society of Japan - J PHYS SOC JPN*, 74:2966–2972.

Seiji Miyoshi, Kazuyuki Hara, and Masato Okada. 2005. Analysis of ensemble learning using simple perceptrons based on online learning theory. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 71:036116.

Rui Xia, Chengqing Zong, and Shoushan Li. 2011a. Ensemble of feature sets and classification algorithms for sentiment classification. *Information Sciences*, 181(6):1138 – 1152.

Rui Xia, Chengqing Zong, and Shoushan Li. 2011b. Ensemble of feature sets and classification algorithms for sentiment classification. *Inf. Sci.*, 181:1138–1152.

CAiRE_HKUST at SemEval-2019 Task 3: Hierarchical Attention for Dialogue Emotion Classification

Genta Indra Winata*, Andrea Madotto*, Zhaojiang Lin, Jamin Shin, Yan Xu, Peng Xu, Pascale Fung
Center for Artificial Intelligence Research (CAiRE)
Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong
{giwinata,amadotto,zlinao}@connect.ust.hk,
{jmshinaa,yxucb,pxuab}@connect.ust.hk,pascale@ece.ust.hk

Abstract

Detecting emotion from dialogue is a challenge that has not yet been extensively surveyed. One could consider the emotion of each dialogue turn to be independent, but in this paper, we introduce a hierarchical approach to classify emotion, hypothesizing that the current emotional state depends on previous latent emotions. We benchmark several feature-based classifiers using pre-trained word and emotion embeddings, state-of-the-art end-to-end neural network models, and Gaussian processes for automatic hyper-parameter search. In our experiments, hierarchical architectures consistently give significant improvements, and our best model achieves a 76.77% F1-score on the test set.

1 Introduction

Customer service can be challenging for both the givers and receivers of services, leading to emotions on both sides. Even human service-people who are trained to deal with such situations struggle to do so, partly because of their own emotions. Neither do automated systems succeed in such scenarios. *What if we could teach machines how to react under these emotionally stressful situations of dealing with angry customers?*

This paper represents work on the SemEval 2019 shared task (Chatterjee et al., 2019b), which aims to bring more research on teaching machines to be empathetic, specifically by contextual emotion detection in text. Given a textual dialogue with two turns of context, the system has to classify the emotion of the next utterance into one of the following emotion classes: Happy, Sad, Angry, or Others. The training dataset contains 15K records for emotion classes, and contains 15K records not belonging to any of the aforementioned emotion classes.

The most naive first step would be to recognize emotion from a given flattened sequence, which has been researched extensively despite the very abstract nature of emotion (Socher et al., 2013; Felbo et al., 2017a; McCann et al., 2017; Xu et al., 2018; Chatterjee et al., 2019a). However, these *flat* models do not work very well on dialogue data as we have to merely concatenate the turns and flatten the hierarchical information. Not only does the sequence get too long, but the hierarchy between sentences will also be destroyed (Hsu and Ku, 2018; Kim et al., 2018). We believe that the natural flow of emotion exists in dialogue, and using such hierarchical information will allow us to predict the last utterance's emotion better.

Naturally, the next step is to be able to detect emotion with a hierarchical structure. To the best of our knowledge, this task of extracting emotional knowledge in a hierarchical setting has not yet been extensively explored in the literature. Therefore, in this paper, we investigate this problem in depth with several strong hierarchical baselines and by using a large variety of pre-trained word embeddings.

2 Methodology

In this task, we focus on two main approaches: 1) feature-based and 2) end-to-end. The former compares several well-known pre-trained embeddings, including *GloVe* (Pennington et al., 2014), *ELMo* (Peters et al., 2018), and *BERT* (Devlin et al., 2018), as well as emotional embeddings. We combine these pre-trained features with a simple Logistic Regression (LR) and XGBoost (Chen and Guestrin, 2016) model as the classifier to compare their effectiveness. The latter approach is to train

*Equal contribution.

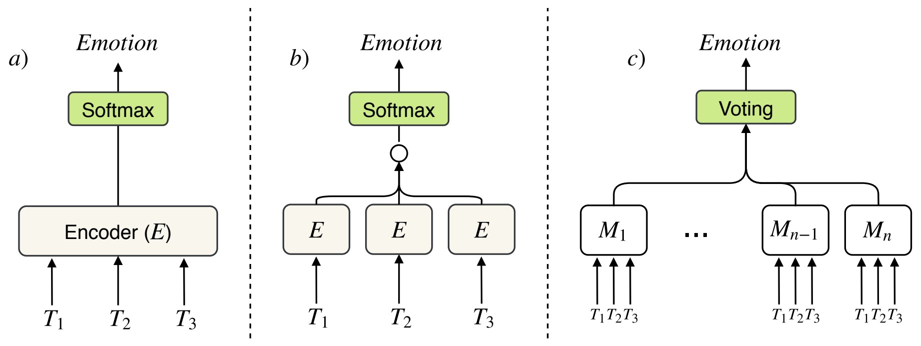


Figure 1: a) Flat model; b) Hierarchical model; c) Voting scheme

a model fully end-to-end with back-propagation. We mainly compare the performances of *flat* models and *hierarchical* models, which also take into account the sequential turn information of dialogues.

2.1 Feature-based Approach

The pre-trained feature-based approach can be subdivided into two categories: 1) word embeddings pre-trained only on semantic information, and 2) emotional embeddings that augment word embeddings with emotional or emoji information. We also examine the use of both categories.

Word Embeddings These include the standard pre-trained non-contextualized GloVe (Pennington et al., 2014), the contextualized embeddings from the bidirectional long short term memory (biLSTM) language model ELMo (Peters et al., 2018), and the more recent *transformer* based embeddings from the bidirectional language model BERT (Devlin et al., 2018).

Emotional Embeddings These refer to two types of features equipped with emotional knowledge. The first is a word-level emotional representation called Emo2Vec (Xu et al., 2018). It is trained with six different emotion-related tasks and has shown extraordinary performance over 18 different datasets. The second is a sentence-level emotional representation called DeepMoji (Felbo et al., 2017b), trained with a biLSTM with an attention model to predict emojis from text on a 1,246 million tweet corpus. Finally, we use Emoji2Vec (Eisner et al., 2016) which directly maps emojis to continuous representations.

2.2 End-to-End Approach

We consider four main models for the end-to-end approach: fine-tuning ELMo (Peters et al., 2018), fine-tuning BERT (Devlin et al., 2018), Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), and Universal Transformer (UTRS) (Dehghani et al., 2018).[1] In the latter model, we also run a Gaussian process for automatic hyper-parameter selection.

ELMo This model from Peters et al. (2018) is a deep contextualized embedding extracted from a pre-trained bidirectional language model that has shown state-of-the-art performance in several natural language processing (NLP) tasks.

BERT This is the state-of-the-art bidirectional pre-trained language model that has recently shown excellent performance in a wide range of NLP tasks. Here, we use $BERT_{BASE}$[2] as our sentence encoder. However, the original model failed to capture the emoji features due to the fact that all the emoji tokens are missing in the vocab. Therefore, we concatenate each sentence representation from BERT with bag of words *Emoji2Vec* (Eisner et al., 2016). Then, a UTRS is used as a context encoder to encode the whole sequence.

LSTM and Universal Transformer LSTM is the widely known model used almost ubiquitously in the literature, while UTRS is a recently published recurrent extension of the multi-head self-attention based model, Transformer from (Vaswani et al., 2017). Finally, for all models, we consider a hierarchical extension which considers the turn information as well. We add another instance of the same model to also encode sentence-level information on top of the word-level representations. We also apply word-level attention to

[1]We also tested Transformer, but had an overfitting issue
[2]We used a PyTorch implementation from https://github.com/huggingface/pytorch-pretrained-BERT

Table 1: The table shows the F1 score on LR and XGBoost.

Feature(s)	Classifier	F1
DeepMoji	LR	64.87
ELMo	LR	63.86
GLoVe	LR	55.11
Emo2Vec	LR	50.91
BERT	LR	44.51
Emoji2Vec	LR	30.45
ELMo + DeepMoji	**LR**	**65.63**
ELMo + Emo2Vec	LR	65.42
Emoji2Vec + GLoVe	LR	58.00
ELMo + DeepMoji	**XGBoost**	**69.86**

Table 2: The table shows F1 score on flat and hierarchical end-to-end models. GP denotes as Gaussian process.

Model	Flat	Hierarchical
LSTM	72.53	73.45
LSTM+GLoVe	**73.95**	**75.64**
LSTM+GLoVe+Emo2Vec	73.85	74.59
UTRS	72.41	74.06
ELMo	68.14	70.55
BERT	66.12	73.29

Table 3: The table shows F1 score on different ensemble models. XGB denotes XGBoost with ELMo and DeepMoji features. ALL denotes all ensemble models.

Model	F1
Ensemble$_1$ (3 HLSTMs)	76.08
Ensemble$_2$ (HBERT + HLSTM + HUTRS)	75.76
Ensemble$_3$ (HBERT + 3 HLSTMs + HUTRS)	76.26
Ensemble$_4$ (HBERT + 5 HLSTMs + HUTRS)	76.24
Ensemble$_5$ (HBERT + 5 HLSTMs + HUTRS)	76.20
Ensemble$_{final}$ (ALL + HLSTM + XGB)	**76.77**
- Angry	75.88
- Happy	73.65
- Sad	81.30

select the important information words on each dialogue turn.

3 Evaluation

In this section, we present the evaluation metrics used in the experiment, followed by results on feature-based, end-to-end, and ensemble approaches and Gaussian process search.

3.1 Training Details

Feature-Based For the feature-based approach, we run LR and XGBoost on features using the Scikit-Learn toolkit (Pedregosa et al., 2011) without any additional tuning.

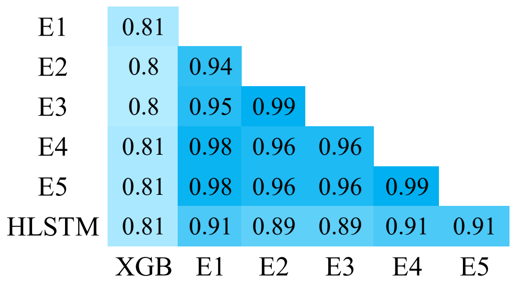


Figure 2: Pearson correlation matrix of a model to other models. E1–E5 denote ensemble models.

ELMo For the flat model, we pre-train ELMo by only fine-tuning the scalar-mix weights, as suggested in Peters et al. (2018). We extract a 1024-dimension bag-of-words representation for each turn and concatenate the three turns into a 3072-dimension vector which is passed to a multilayer perceptron (MLP). For the hierarchical model, we employ two methods: 1) run an LSTM model over each turn's representation 2) pre-extract all three layer weights (LSTM and CNN) and concatenate them into a 3072-dimension vector representation for each turn, which is then passed to an LSTM model. We report the results of the latter pre-extracted method as it performs better.

BERT For the implementation details of $BERT_{BASE}$, we refer interested readers to Devlin et al. (2018). Note that for hierarchical BERT, we use a six-layer UTRS as the context encoder. Each layer of UTRS consists of a multi-head attention block with four heads, where the dimension of each head is set to be ten, and a convolution feed forward block with 50 filters. We use modified Adam optimizer from Devlin et al. (2018) to train our model. The initial learning rate and dropout are 5e-5 and 0.3 respectively.

LSTM and Universal Transformer We train hierarchical LSTMs with hidden sizes of {1000, 1500} using different dropouts {0.2,0.3,0.4,0.5}. The best LSTMs (without additional features, with GLoVE, with GLoVE+Emo2Vec) reported in Figure 2 have a hidden size of 1000 and dropout of 0.5, a hidden size of 1500 and dropout of 0.2, and a hidden size of 1000 and dropout of 0.4 respectively. Then, we train the UTRS using the best hyper-parameters found by the GP. It has a hidden size of 488 with a single hop and ten attention multi-heads. Noam (Vaswani et al., 2017) is used as the learning rate decay.

Gaussian Processes GP hyper-parameter search returns a set of hyper-parameters, both continuous and discrete, and it returns the validation set F1 score. We implement the GP model using an existing library called GPyOpt.[3] We run a GP for 100 iterations using the Expected Improvement (Jones et al., 1998) acquisition function with 0.05 jitter as a starting point. We use a hierarchical universal transformer (HUTRS) as the base model since is the model with the most hyper-parameters to tune with a single split.

3.2 Evaluation Metrics

The task is evaluated with a micro F1 score for the three emotion classes, i.e., Happy, Sad and Angry, and by taking the harmonic mean of the precision and the recall. This scoring function has been provided by the challenge organizers (Chatterjee et al., 2019b).

3.3 Voting Scheme

For each model, we randomly shuffle and split the training set ten times and we apply a voting scheme to create a more robust prediction. We use a majority vote scheme to select the most often seen predictions, and in case of ties, we give the priority to *Others*. This scheme is applied to all end-to-end models since it improved the validation set performance.

3.4 Ensemble Models

To further refine our predictions, we build ensembles of different models. We create five ensemble models by combining the hierarchical version of BERT, LSTM, and UTRS. Finally, we gather two lesser-performing models, a hierarchical LSTM and the best feature-based model (XGBoost with ELMo and DeepMoji features), and we combine them with five ensemble predictions using majority voting to get our final prediction. Finally, we show the Pearson correlation between models in Figure 2.

3.5 Experimental Results

From Table 1, we can see that the DeepMoji features outperforms all the other features by a large margin. Indeed, DeepMoji has been trained using a large emotion corpus, which is compatible with the current task. Emoji2Vec get a very low F1-score since it includes only emojis, and indeed, by adding GLoVe, a more general embedding, we achieve better performance. For the end-to-end approach, hierarchical biLSTM with GLoVe word embedding achieves the highest score with a 75.64% F1-score. Our ensemble achieves a higher score compared to individual models. The best ensemble model achieves a 76.77% F1-score. As shown in Table 3, the ensemble method is effective to maximize the performance from a bag of models.

4 Related work

Emotional knowledge can be represented in different ways. Word-level emotional representations, inspired from word embeddings, learn a vector for each word, and have shown effectiveness in different emotion related tasks, such as sentiment classification (Tang et al., 2016), emotion classification (Xu et al., 2018), and emotion intensity prediction (Park et al., 2018). Sentence-level emotional representations, such as DeepMoji (Felbo et al., 2017a), train a biLSTM model to encode the whole sentence to predict the corresponding emoji of the sentence. The learned model achieves state-of-the-art results on eight datasets. Sentiment lexicons from Taboada et al. (2011) show that word lexicons annotated with sentiment/emotion labels are effective in sentiment classification. This method is further developed using both supervised and unsupervised approaches in Wang and Xia (2017). Also, other models, such as a deep averaging network (Iyyer et al., 2015), attention-based network (Winata et al., 2018), and memory network (Dou, 2017), have been investigated to improve the classification performance. Practically, the application of emotion classification has been investigated on interactive dialogue systems (Bertero et al., 2016; Winata et al., 2017; Siddique et al., 2017; Fung et al., 2018).

5 Conclusion

In this paper, we compare different pre-trained word embedding features by using Logistic Regression and XGBoost along with flat and hierarchical architectures trained in end-to-end models. We further explore a GP for faster hyper-parameter search. Our experiments show that hierarchical architectures give significant improvements and we further gain accuracy by combining the pre-trained features with end-to-end models.

[3]http://sheffieldml.github.io/GPyOpt/

References

Dario Bertero, Farhad Bin Siddique, Chien-Sheng Wu, Yan Wan, Ricky Ho Yin Chan, and Pascale Fung. 2016. Real-time speech emotion and sentiment recognition for interactive dialogue systems. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1042–1047.

Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. 2019a. Understanding emotions in text using deep learning and big data. *Computers in Human Behavior* 93:309–317.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019b. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Minneapolis, Minnesota.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, KDD '16, pages 785–794. https://doi.org/10.1145/2939672.2939785.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819* .

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Zi-Yi Dou. 2017. Capturing user and product information for document level sentiment analysis with deep memory network. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 521–526.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bosnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. In *Proceedings of The Fourth International Workshop on Natural Language Processing for Social Media*. pages 48–54.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017a. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 1615–1625.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017b. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *2017 Conference on Empirical Methods in Natural Language ProcessingEmpirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Pascale Fung, Dario Bertero, Peng Xu, Ji Ho Park, Chien-Sheng Wu, and Andrea Madotto. 2018. Empathetic dialog systems.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Chao-Chun Hsu and Lun-Wei Ku. 2018. Socialnlp 2018 emotionx challenge overview: Recognizing emotions in dialogues. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*. pages 27–31.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 1681–1691.

Donald R Jones, Matthias Schonlau, and William J Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global optimization* 13(4):455–492.

Yanghoon Kim, Hwanhee Lee, and Kyomin Jung. 2018. Attnconvnet at semeval-2018 task 1: Attention-based convolutional neural networks for multi-label emotion classification. In *Proceedings of The 12th International Workshop on Semantic Evaluation*. pages 141–145.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*. pages 6294–6305.

Ji Ho Park, Peng Xu, and Pascale Fung. 2018. Plusemo2vec at semeval-2018 task 1: Exploiting emotion knowledge from emoji and# hashtags. In *Proceedings of The 12th International Workshop on Semantic Evaluation*. pages 264–272.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. volume 1, pages 2227–2237.

Farhad Bin Siddique, Onno Kampman, Yang Yang, Anik Dey, and Pascale Fung. 2017. Zara returns: Improved personality induction and adaptation by an empathetic virtual agent. *Proceedings of ACL 2017, system demonstrations* pages 121–126.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1631–1642. http://www.aclweb.org/anthology/D13-1170.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics* 37(2):267–307.

Duyu Tang, Furu Wei, Bing Qin, Nan Yang, Ting Liu, and Ming Zhou. 2016. Sentiment embeddings with applications to sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering* 28(2):496–509.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. pages 5998–6008.

Leyi Wang and Rui Xia. 2017. Sentiment lexicon construction with representation learning based on hierarchical sentiment supervision. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 502–510.

Genta Indra Winata, Onno Kampman, Yang Yang, Anik Dey, and Pascale Fung. 2017. Nora the empathetic psychologist. *Proc. Interspeech 2017* pages 3437–3438.

Genta Indra Winata, Onno Pepijn Kampman, and Pascale Fung. 2018. Attention-based lstm for psychological stress detection from spoken language using distant supervision. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pages 6204–6208.

Peng Xu, Andrea Madotto, Chien-Sheng Wu, Ji Ho Park, and Pascale Fung. 2018. Emo2vec: Learning generalized emotion representation by multi-task training. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. pages 292–298.

CECL at SemEval-2019 Task 3: Using Surface Learning for Detecting Emotion in Textual Conversations

Yves Bestgen
Centre for English Corpus Linguistics
Université catholique de Louvain
Place Cardinal Mercier, 10 1348 Louvain-la-Neuve Belgium
yves.bestgen@uclouvain.be

Abstract

This paper describes the system developed by the Centre for English Corpus Linguistics for the SemEval-2019 Task 3: EmoContext. It aimed at classifying the emotion of a user utterance in a textual conversation as happy, sad, angry or other. It is based on a large number of feature types, mainly unigrams and bigrams, which were extracted by a SAS program. The usefulness of the different feature types was evaluated by means of Monte-Carlo resampling tests. As this system does not rest on any deep learning component, which is currently considered as the state-of-the-art approach, it can be seen as a possible point of comparison for such kind of systems.

1 Introduction

This paper presents the participation of the Centre for English Corpus Linguistics (CECL) in the SemEval-2019 Task 3 "EmoContext: Contextual Emotion Detection in Text". The objective of the task is to classify the emotion of a user utterance in a textual conversation with a bot as happy, sad, angry or other. Contextual information is provided by means of the two previous utterances, the one just before, which was produced by the bot, and the first one of the triplet produced by the same person as the third. This task can be seen as a difficult problem in absence of any non-written information (Gupta et al., 2017).

Because of its recent and important development for the analysis of big data and especially for natural language processing, deep learning has become the preferred procedure to take up this kind of challenge (Chatterjee et al., 2019a). However, detecting emotion in texts is a well-established field of research for which unsupervised approaches have been proposed in content analysis (e.g. Anderson and McMaster, 1982; Bestgen, 1994) as well as less recent supervised approaches such as SVM (Chatterjee et al., 2019a; Pang and Lee, 2008). SVM has long been considered as the state-of-the-art in text categorization (Joachims, 2002). Therefore, it seemed interesting to get an idea of its effectiveness for the present task in comparison to deep learning approaches that should be used by many participants in this challenge. Trying to determine what level of performance can be achieved with a surface learning system was thus the main focus of this study. It should be noted that the developed system does not rely on complementary training data, nor on lexical emotional norms produced manually or automatically (Bestgen and Vincze, 2012), nor on semantic knowledge extracted from large databases or large corpora (Miller et al., 1990). However, several attempts have been made to increase its effectiveness by adding more complex features to the usual token n-grams.

The remainder of this report describes the datasets made available for this challenge, the systems developed, and the results obtained as well as the analyses performed to get a better idea of the factors that affect the system performance as well as the usefulness of the various types of features used.

2 Data

The challenge organizers divided the materials into three datasets (see Chatterjee et al. (2019b) for details):

- The learning set (Learn) that contained 30160 instances of which approximately 16.7% of the three emotion categories and the remaining 50% of Other,
- The development set (Dev) that contained 2755 instances of which approximately 5% of the emotion categories and the remaining 85% of Other

- The test set (Test) that contained twice as many instances as the Dev set and the same proportion of the four categories as in that set.

The true labels for the Learn set were available from the beginning of the training phase, those for the Dev set on December 10 and those for the Test set after the end of the challenge. Between the beginning of the development phase (August 21, 2018) and the beginning of the test phase (January 18, 2019), it was possible to use Codalab to evaluate the systems on the Dev set.

3 Systems

This section describes the systems developed to maximize the approach effectiveness. The feature extraction was performed by means of a custom SAS program running in SAS University (freely available for research at http://www.sas.com/en_us/software/university-edition.html). The predictive models used during the development phase were built on the Learn set and evaluated on the Dev set (see section 4 for a justification) by means of the L1-regularized L2-loss Support Vector Machine for classification (L1-L2-SVM) available in the LIBLINEAR package (-s 5, Fan et al., 2008).

3.1 Base System

The main part of the system consists of tokens and tokens n-grams present in the three utterances of an instance to be categorized. First, the utterances are processed by a Perl script that group the most frequent contracted forms with their corresponding full forms (e.g. I'm > I am). Then, an ad-hoc tokenizer splits each sequence of characters separated by a space in tokens composed of (lowercased) letters, numbers, punctuation marks or emojis, trying not to cut the potential emoticons such as :-) and :D. In the case of tokens composed exclusively of alphabetic characters, the encoder detects the presence of the same character at least three times at the end of the token (e.g. ahhh) and keeps only one occurrence.

The features for the SVM were generated from these tokens. These are mainly unigrams and bigrams of tokens, present in each utterance, to which a tag is added to keep trace of the utterance it comes from. The tokens, bigrams, and trigrams from the first and the last utterances, those produced by the human participant, are also outputted without being distinguished by a tag so that their frequencies add up. The emojis of the first and third utterances are also processed specifically. Each token composed of several emojis is not only produced as it is, but the different emojis that compose it are also outputted separately.

Finally, a feature frequency dictionary based on all the available materials, thus also on the Test set, is produced in order to be able to use a minimum frequency threshold (set in all the analyses reported here at 2). This dictionary is also used to number the features in order to put them in LIBLINEAR format. The weight of each feature is equal to the logarithm in base 10 of its frequency in the instance to which one is added.

3.2 Extended System

The basic system was extended during the development period with a series of meta-features that seemed to improve its predictive power. They consist in a dozen simple global statistics computed on each of the utterances: number of tokens, number of characters without the spaces, number of capital letters and a potential index of emotional intensity based on the number of characters repeated at least three times, the number of emoticons and of emojis (whatever their meaning). These statistics were divided by the maximum values observed in the dataset.

Another addition was made specifically for the instances whose third utterance contained only the "yes" token: the content of the first utterance was copied to this third one.

3.3 Parameters

The regularization meta-parameter C of the SVM was optimized using a grid search. Although LIBLINEAR has a built-in program for optimizing C, we used our optimization program to have more flexibility in choosing the values to test. To take into account the differences between the frequencies of the four categories in the Learn and the other two datasets, the LIBLINEAR -wi parameter, which allows modulating the C parameter according to the category, was used. The values were set using a heuristic approach.

4 Analyses and Results

All the results reported below are expressed in terms of the challenge metric, which is the microaveraged F1 score (F1μ) for the three emotion classes (see Chatterjee et al. (2019b) for details). A first analysis was aimed to determine whether it was possible to extract several development datasets from the Learn set that produced similar performances to those obtained by using the Dev set provided by the organizers so as to limit the risk of overfitting when only one evaluation dataset is used. To this end, ten development samples were independently extracted from the Learn set in such way that they contained exactly the same number of instances in each of the four categories as in the official Dev set[1]. The results showed that the performances on these ten samples were systematically much higher than those obtained on the Dev set (0.78 vs. 0.69 for the version of the system available at that time). It would thus seem that the Dev set has specificities that distinguish it quite clearly from the Learn set and therefore all the developments have been made on the Dev set, accepting the risk of overfitting.

4.1 Official Performance on the Test Set

During the test phase, teams were allowed to submit 30 trials during 9 days. The base system (System 1) was first submitted to Codalab and obtained a F1μ of 0.721[2]. The remaining 29 trials were used to try improving performance. Three tracks were followed:

- First, I tried to optimize the number of instances assigned to each category in order to obtain better recall and precision values. To this end, I used the L1-regularized logistic regression (L1-LR) available in LIBLINEAR (-s 6) which is for the present datasets a little less effective than the L1-L2-SVM used previously but is able to estimate the probabilities of each instance belonging to each class. The most effective approach found was to classify into a category the same number of instances as the one actually presents in the Test set (the needed counts were obtained during the test phase by means of the procedure described in Note 1). This system 2 obtained a F1μ of 0.726.
- Then, the predictions of these two first systems were combined to try maximizing the F1s in each category because it was observed that System 1 was less accurate than System 2 for the Angry and Sad categories, but more accurate for the Happy category. Since the prediction differences between the two systems were systematically changes from one emotion category to the Other category or vice versa, but never from one emotion category to another one, it was easy to combine the two systems by taking the Happy predictions of System 1 and the Sad and Angry predictions of System 2 and placing the remaining instances in the Other category. This System 3 reached a F1μ of 0.73.
- Finally, System 3 was combined with a model based only on the third utterance. This model was used to move to the Other category instances assigned to one of the three emotional categories for which the probability of membership (provided by the logistic regression) was the lowest. A series of tests carried out on Codalab to optimize the decision rules made it possible to reach the final system performance of 0.736 (F1_Angry=0.7331, F1_Happy=0.7035, F1_Sad=0.7746).

4.2 Factors that Affect the System Performance

The remainder of this report analyzes the impact of several factors, including the different types of features, to the system performance. All these analyses were conducted using various versions of System 1. They aimed at categorizing the Dev set using the Learn set to build the model and the Test set using the Learn and Dev sets to build the model. In these analyses, the L1-L2 SVM was used with the parameters considered as optimal for each evaluation dataset. In order to determine if the observed differences were statistically significant, two Monte-Carlo resampling tests (Howell, 2008, Chap. 18) were used, a test for related samples to compare two different models on the same evaluation set and a test for independent samples to compare the performance on the Dev and Test sets.

A first analysis showed that using only the Learn set for predicting the Test set instead of the concatenation of the Learn and Dev sets hurt the performance (0.716 vs. 0.721), but the dif-

[1]As these analyses were performed before the labels for the Dev set had been released, I used the number of instances assigned to each category in a submission and the precision and recall for each category outputted by CodaLab for that submission to calculate the actual category frequencies.

[2]SVM parameters for this system were as following: $C = 0.67$, $w\_angry = 0.275$, $w\_happy = 0.31$ and $w\_sad = 0.275$.

System description	Dev Set F1μ	Dev Set Diff.	Dev Set p	Test Set F1μ	Test Set Diff.	Test Set p
Full (System 1)	0.762			0.721		
Without Meta-Features	0.752	-0.010	0.018	0.719	-0.002	0.715
Without 3grams	0.752	-0.010	0.007	0.719	-0.002	0.621
Without 2grams and 3grams	0.735	-0.027	0.002	0.710	-0.011	0.107
Without Utterance 2	0.751	0.012	-0.041	0.726	+0.005	0.193
Without Emojis Processing	0.746	-0.016	0.010	0.709	-0.012	0.006
Without Repeated Letters Processing	0.760	-0.002	0.508	0.722	+0.001	0.411

Table 1: F1μ, difference from the full system, and p-value for the ablation approach.

ference was far from being statistically significant ($p > 0.30$).

A second analysis showed that the system was more efficient on the Dev set (0.762) than on the Test set (0.721), a statistically significant difference ($p = 0.043$). Providing that the Dev and Test sets were randomly extracted from the same sample, this result suggests that the system suffers from overfitting since it was developed on the basis of the Dev set.

The ablation approach was then used to assess the independent contribution of each type of features to the overall performance. It consists in removing some sets of features of the model and reevaluating it. As Table 1 shows, the impact of removing feature types was often quite different on the two evaluation sets. Suppressing most of the feature types for the Dev set produced a sizable decrease in performance while a very small decrease or even an increase in performance was observed for the Test set. Significance tests, which compare the ablated systems to the full one, confirm this analysis. Table 1 also shows that the specific treatment of emojis seemed particularly useful for both datasets.

A final analysis consisted in constructing a system from which all feature types in Table 1 were simultaneously removed except the specific treatment of emojis. This system obtained a F1μ of 0.743 on the Dev set, a decrease of 0.019, and a F1μ of 0.731 on the Test set, an increase of 0.01.

5 Conclusion

The main goal of this study was to try to determine what level of performance could be reached in the EmoContext task by employing a non-deep learning approach. The achieved F1μ was between 0.721 and 0.736, ranking the system approximately in the first quarter of the 165 teams who submitted a prediction for the Test set, yet far enough from the top teams who achieved a F1μ of 0.79. Looking at the other system description papers should allow to find out whether other teams used a surface learning approach while achieving better performance.

To conclude, it seems useful to underline an observation reported above that could have greatly affected the performance of the system, but also of other systems. The analyses carried out showed that the Dev set provided by the organizers gave rise to significantly lower performances than those obtained by using evaluation datasets from the Learn set as similar as possible to the actual Dev set. This led me to develop the system on the basis of the Dev set provided by the organizers and the consequence was a statistically significant overfit on the Dev set when compared to the Test set. In the absence of information about how the three datasets were created, it is not possible to comment on the origin of these differences.

Acknowledgments

The author wishes to thank the organizers of the EmoContext task for putting together this valuable event. The author is a Research Associate of the Fonds de la Recherche Scientifique - FNRS.

References

C. W. Anderson and G. E. McMaster. 1982. Computer assisted modeling of affective tone in written documents. *Computers and the Humanities*, 16(1):1–9.

Yves Bestgen. 1994. Can emotional valence in stories be determined from words? *Cognition and Emotion*, 8(1):21–36.

Yves Bestgen and Nadja Vincze. 2012. Checking and bootstrapping lexical norms by means of word similarity indexes. *Behavior Research Methods*, 44(4):998–1006.

Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. 2019a. Understanding emotions in text using deep learning and big data. *Computers in Human Behavior*, 93:309–317.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019b. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, USA.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations. *arXiv preprint arXiv:1707.06996*.

David Howell. 2008. *Méthodes statistiques en sciences humaines*. De Boeck Université, Bruxelles.

Thorsten Joachims. 2002. *Learning to Classify Text Using Support Vector Machines – Methods, Theory, and Algorithms*. Kluwer/Springer.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. 1990. Introduction to WordNet: An On-line Lexical Database*. *International Journal of Lexicography*, 3(4):235–244.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1):1–135.

CLaC Lab at SemEval-2019 Task 3: Contextual Emotion Detection Using a Combination of Neural Networks and SVM

Elham Mohammadi, Hessam Amini and Leila Kosseim
Computational Linguistics at Concordia (CLaC) Lab
Department of Computer Science and Software Engineering
Concordia University, Montréal, Québec, Canada
first.last@concordia.ca

Abstract

This paper describes the *CLaC Lab* system at SemEval 2019, Task 3 (*EmoContext*), which focused on the contextual detection of emotions in a dataset of 3-round dialogues. For our final system, we used a neural network with pretrained *ELMo* word embeddings and POS tags as input, GRUs as hidden units, an attention mechanism to capture representations of the dialogues, and an SVM classifier which used the learned network representations to perform the task of multi-class classification. This system yielded a micro-averaged F1 score of 0.7072 for the three emotion classes, improving the baseline by approximately 12%.

1 Introduction

Automatic emotion detection has been the focus of much research in a variety of fields, including emotion detection based on images (Rao et al., 2019), speech signals (Davletcharova et al., 2015), electroencephalography (EEG) signals (Ackermann et al., 2016), and texts (Tafreshi and Diab, 2018).

With the advent of social media, emotion detection from text has been used to track bloggers' mental health and has been explored using different techniques, such as lexicon-based approaches and machine learning (Canales and Martínez-Barco, 2014). Lexicon-based approaches include keyword-based and ontological approaches. In a keyword-based approach (e.g. Ma et al., 2005), a specific set of opinion terms and their WordNet synonyms and antonyms are used to determine the emotion class of the text. On the other hand, ontology-based approaches (e.g. Sykora et al., 2013) try to detect emotions by taking into account the knowledge of concepts, the interconnection between them, and their final emotional impact. To achieve better generalization, the resources used in lexicon-based approaches can be employed as input features to supervised or unsupervised machine learning models.

The drawback with supervised machine learning approaches lies in the need for a large corpus of labelled data. Abdul-Mageed and Ungar (2017) collected a labelled twitter dataset of 1/4 billion tweets, spanning over 8 primary emotions, using distant supervision, i.e. collecting tweets with emotion hashtags that can be used as labels. Then using a gated recurrent neural network for classification, they achieved an accuracy of 95.68%, superior to the best previously published results by Volkova and Bachrach (2016).

Although much research has focused on the detection of emotions in tweets and blog posts (e.g. Mohammad, 2012; Desmet and Hoste, 2013; Liew and Turtle, 2016), emotion detection in dialogues, as well as in single utterances has received very little attention. This topic can have significant impact for the development of social chatbots, with the aim of creating an emotional connection between a user and a chatbot (Banchs, 2017). Task 3 of SemEval 2019 (*EmoContext*) has focussed on contextual emotion detection over 4 classes: *happy*, *sad*, *angry*, and *others* (Chatterjee et al., 2019b). We participated in this shared task under the name *CLaC Lab* and used a combination of artificial neural networks (with recurrent units and attention mechanism) and Support Vector Machines (SVM) to address this multi-class classification task.

The rest of the paper is organized as follows: Section 2 describes the overall methodology. Section 3 presents a detailed explanation of the different components of the system. Section 4 presents the results of the system. Section 5 discusses some interesting findings from our participation to this task. Finally, Section 6 is dedicated to the conclusion of this work.

2 Methodology

This section presents the overall methodology used to perform the task of contextual emotion detection. An overview of the system architecture is presented; while more detailed explanation can be found in Section 3.

2.1 Neural Architecture

The core of our system is a neural network that is trained to learn the feature representations necessary to train our final classifier. Figure 1 shows the overall architecture of the neural network that we used.

The Input As shown in Figure 1, each input sample consists of three consecutive utterances of a dialogue between two interlocutors. We consider each utterance as a sequence of tokens (words). As a result, each utterance is represented as a vector, such as $[x_{i,1}, x_{i,2}, \dots, x_{i,t}, \dots, x_{i,n}]$, where $x_{i,t}$ is the vector representation of the t-th word in the i-th utterance, and n is the length of the i-th uttermace.

The Recurrent Component The input layer is followed by a bidirectional hidden recurrent component. Each utterance is fed to a separate hidden component, who is responsible to process that specific utterance in a forward and backward pass.

For the forward pass, the content value of the hidden component at a specific time-step, h_t, relies on both the value of the current input, x_t, and the content value of the hidden component itself at the previous time-step, h_{t-1} (Equation 1). The content value produced at this stage is then passed through another mapping function, f_y, which generates the output value of the hidden component at the current time-step, y_t (Equation 2).

$$h_t = f_h(x_t, h_{t-1}) \tag{1}$$

$$y_t = f_y(h_t) \tag{2}$$

The backward pass differs from the forward pass, in that in Equation 1, h_{t-1} is replaced by h_{t+1}, meaning that the content value of the hidden component relies on the content value at its next time-step instead of the previous one. The output calculation is identical to the forward pass (see Equation 2).

Attention Mechanism Vaswani et al. (2017) describes an attention mechanism as the weighted sum of several values (i.e. vectors), where the weight assigned to each value can be computed using a compatibility function. Using this description, the overall function of the attention mechanism for our task can be defined using Equation 3, where $\omega(y_{t'})$ refers to the weight assigned to the output of the hidden layer at time-step t', and N is the number of time-steps (i.e. the length of the utterance).

$$attn = \sum_{t'=1}^{N} y_{t'}\omega(y_{t'}) \tag{3}$$

Although originally developed for the task of machine translation (Bahdanau et al., 2014), attention mechanisms have been shown to significantly improve text classification tasks (e.g. Yang et al., 2016; Zhou et al., 2016; Wang et al., 2016; Cianflone et al., 2018). Following these works, we used attention in our system (see Figure 1).

Classification Once the neural network has created a representation of the input, a final feed-forward classification network, which takes as input the concatenated vectors from the attention units of the three utterances, performs the classification task.

2.2 Support Vector Classifier

The neural network was not used to do the final classification, but was used only as a feature extractor. This is illustrated in Figure 1 by the dotted connections between the attention units and the classifier. The extracted features were fed to an SVM (Cortes and Vapnik, 1995), which acted as the classifier.

Our main drive for using an SVM was due to explicit handling of margin size versus misclassification rate (i.e. variance versus bias). This, alongside the deterministic nature of an SVM and its faster training process (in comparison to a neural network), enabled us to play with its several configurations in order to find the optimal one for our task.

3 System Overview

In this section, we provide detailed information on the final system's architecture.

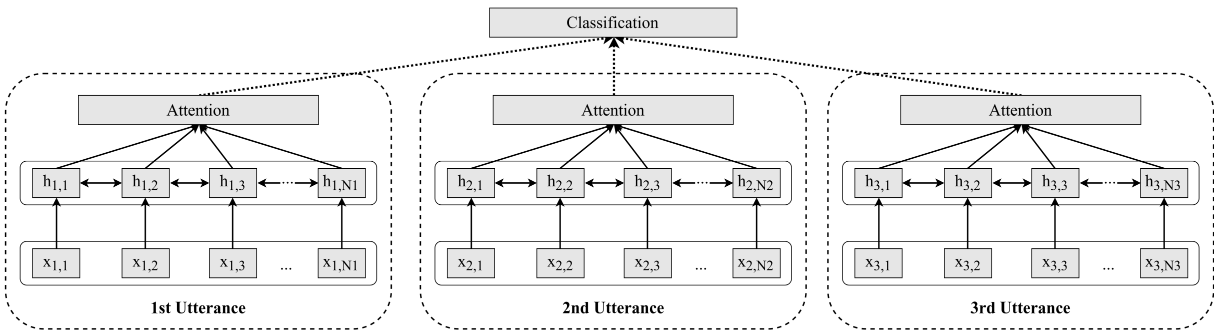


Figure 1: The overall framework of the neural network model.

3.1 The Neural Network

We developed the neural network using *PyTorch* (Paszke et al., 2017). Detailed explanations of the neural network's architecture are provided below.

Input Features Each word in each utterance is sent to the neural network by using their word embeddings, concatenated to their one-hot part-of-speech (POS) tag.

For word embeddings, we used a pretrained *ELMo* word embedder (Peters et al., 2018), which extracts the embeddings of each token in a textual context from their constituent characters. The suitability of *ELMo* for the current task lies in its ability to take into consideration the context of the tokens when generating the word embeddings, and also the handling of out-of-vocabulary words. We used pretrained *ELMo* embeddings of size 1024.

We followed the Penn Treebank tagset standard (Marcus et al., 1993) for assigning POS tags to each token. For this, we used *spaCy*[1] for tokenization and POS tagging of the data.

Recurrent Unit The system uses the bidirectional Gated Recurrent Unit (GRU) architecture, proposed by Cho et al. (2014), and stacks two layers of 25 bidirectional GRUs for the recurrent component of each utterance.

Attention Layer Equations 4, 5 and 6 show the mechanisms used in the attention layer. Using Equation 4, the weight matrix w is applied to the output of the GRU component at each time-step, y_t, which maps each output vector of the recurrent unit (with the size of 2×25 in our case) to a single value, ν_t. Then, using Equation 5, the weights, ω, are calculated. These will be used to calculate the output of the attention layer in Equation 6 (N represents the number of time-steps for each utterance, i.e. the length of the utterance).

$$\nu_t = y_t \times w \tag{4}$$

$$\omega = Softmax([\nu_1, \nu_2, \nu_3, \ldots, \nu_N]) \tag{5}$$

$$attn = \sum_{t'=1}^{N} \omega_{t'} y_{t'} \tag{6}$$

The Classifier The outputs of the attention layers from the three utterances are concatenated, and fed to a fully-connected feed-forward layer, which uses a softmax activation function at the end. The output of the classifier includes a vector of 4 reals, which represent the estimated probability for each of the 4 classes (*happy*, *sad*, *angry*, and *others*).

Optimization Technique Cross-entropy is used as the loss function and weights are applied to each class proportional to the inverse of number of their samples, in order to handle the unbalanced distribution of the four different classes over the data. The Adam optimizer (Kingma and Ba, 2014) was used and the learning rate was set to 10^{-4}. For computational reasons, minibatches of size 32 were used for training the neural network model, and different sequence sizes was handled by zero-padding.

Experiments showed that, when trained on the training dataset only, the neural network reaches its maximum performance on the development dataset (in our case, the micro-averaged F1 score for the three emotion classes) in approximately 7 epochs.

[1] https://spacy.io/

3.2 The SVM

The *scikit-learn* library (Pedregosa et al., 2011) was used for the SVM, which utilized a polynomial kernel with degree of 4. The γ parameter was set to the inverse of the number of features, which was 1/150 in our case (since the model uses 50 features from each utterance's attention layer, and there were 3 utterances for each sample), and set the penalty parameter C equal to 2.5. To train the SVM, we used the samples from both the training and the development datasets.

4 Results

We tested the system using two different configurations: 1) Using the neural network for feature extraction and classification (*NN*); and 2) Using the neural network for feature extraction and the SVM for classification (*NN+SVM*). We compared the two systems with the baseline system provided by the *EmoContext* organizers (Chatterjee et al., 2019a), which uses a neural network with LSTM units (Hochreiter and Schmidhuber, 1997) in the hidden layer and *GloVe* embeddings (Pennington et al., 2014). Table 1 shows the results from our two models in comparison to the baseline configuration.

System	*angry*	*happy*	*sad*	Micro
NN+SVM	**0.7130**	**0.6667**	**0.7443**	**0.7072**
NN	0.6206	0.6374	0.6800	0.6430
Baseline	N/A	N/A	N/A	0.5868

Table 1: F1 scores on the shared task test dataset for each emotion class and the micro-average. The final system (*NN+SVM*) is highlighted in bold.

The results in Table 1 show that, both our system configurations outperformed the baseline system, while the *NN+SVM* is significantly better than the others. We hypothesize that, given the same set of features, an SVM constitutes a stronger classifier due to its deterministic and more robust nature, and also due to its explicit design to optimize the margin size between different classes.

5 Discussion

Several interesting findings are worthy of discussion:

- The use of LSTMs in the neural network design instead of GRU yielded lower results with the development dataset. We believe that this is because the LSTM model was more prone to overfitting due to a higher number of parameters. Also, since most of the utterances were quite short (5.19 tokens on average), a GRU was enough to capture the necessary information.
- The use of POS tags alongside word embeddings did not help in improving the system performance, but it was helpful in stabilizing the output of the system (i.e. less performance fluctuations during training).
- For the SVM, an increase in the parameter C from its default value of 1 to 2.5 achieved slightly better results. We believe that this is due to the neural based features being quite representative of the final classes, and as a result, more penalty had to be assigned to errors during training as opposed to trying to achieve larger decision margins.

6 Conclusion

This paper presented the system that we developed for our participation to SemEval 2019, Task 3 (*EmoContext*). The task focused on detecting four classes of emotions, *happy*, *sad*, *angry*, and *others* in a dataset consisting of small dialogues between two people.

For this task, we developed a system that used pretrained *ELMo* word embeddings alongside POS tags as input features to a bidirectional GRU, followed by an attention layer and outputting a representation of a sample dialogue. This representation was, in turn, used as input to a final SVM classifier. Using this method, we could significantly outperform the baseline system, and achieved a micro-averaged F1 of 0.7072 for the three emotion classes on the test dataset.

Acknowledgments

We would like to express our gratitude to Parsa Bagherzadeh for insightful discussions throughout the course of our participation to *EmoContext*.

This work was financially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Muhammad Abdul-Mageed and Lyle Ungar. 2017. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 718–728, Vancouver, Canada. Association for Computational Linguistics.

Pascal Ackermann, Christian Kohlschein, Jó Agila Bitsch, Klaus Wehrle, and Sabina Jeschke. 2016. Eeg-based automatic emotion recognition: Feature extraction, selection and classification methods. In *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pages 1–6, Munich, Germany. IEEE.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *Computing Research Repository*, arXiv:1409.0473.

R. E. Banchs. 2017. On the construction of more human-like chatbots: Affect and emotion analysis of movie dialogue data. In *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1364–1367, Kuala Lumpur, Malaysia. IEEE.

Lea Canales and Patricio Martínez-Barco. 2014. Emotion detection from text: A survey. In *Proceedings of the Workshop on Natural Language Processing in the 5th Information Systems Research Working Days (JISIC)*, pages 37–43, Quito, Ecuador. Association for Computational Linguistics.

Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. 2019a. Understanding emotions in text using deep learning and big data. *Computers in Human Behavior*, 93:309–317.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019b. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Andre Cianflone, Yulan Feng, Jad Kabbara, and Jackie Chi Kit Cheung. 2018. Let's do it "again": A first computational approach to detecting adverbial presupposition triggers. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 2747–2755, Melbourne, Australia. Association for Computational Linguistics.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

Assel Davletcharova, Sherin Sugathan, Bibia Abraham, and Alex Pappachen James. 2015. Detection and analysis of emotion from speech signals. *Procedia Computer Science*, 58:91–96.

Bart Desmet and VéRonique Hoste. 2013. Emotion detection in suicide notes. *Expert Systems with Applications*, 40(16):6351–6358.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Computing Research Repository*, arXiv:1412.6980.

Jasy Suet Yan Liew and Howard R. Turtle. 2016. Exploring fine-grained emotion detection in tweets. In *Proceedings of the NAACL Student Research Workshop*, pages 73–80. Association for Computational Linguistics.

Chunling Ma, Helmut Prendinger, and Mitsuru Ishizuka. 2005. Emotion estimation and reasoning based on affective textual interaction. In *First International Conference on Affective Computing and Intelligent Interaction (ASCII 2005)*, pages 622–628, Beijing, China. Springer.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Saif Mohammad. 2012. #emotional tweets. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 246–255. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS 2017 Autodiff Workshop*, Long Beach, California, USA.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2018)*, pages 2227–2237, New Orleans, Louisiana, USA. Association for Computational Linguistics.

Tianrong Rao, Xiaoxu Li, Haimin Zhang, and Min Xu. 2019. Multi-level region-based convolutional neural network for image emotion classification. *Neurocomputing*, 333:429–439.

Martin D. Sykora, Thomas Jackson, Ann O'Brien, and Suzanne Elayan. 2013. Emotive ontology: Extracting fine-grained emotions from terse, informal messages. *IADIS International Journal on Computer Science and Information Systems*, 8(2):106–118.

Shabnam Tafreshi and Mona Diab. 2018. Emotion detection and classification in a multigenre corpus with joint multi-task deep learning. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*, pages 2905–2913, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 5998–6008. Curran Associates, Inc., Long Beach, California, USA.

Svitlana Volkova and Yoram Bachrach. 2016. Inferring perceived demographics from user emotional tone and user-environment emotional contrast. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1567–1578, Berlin, Germany. Association for Computational Linguistics.

Yequan Wang, Minlie Huang, xiaoyan zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 606–615, Austin, Texas, USA. Association for Computational Linguistics.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*, pages 1480–1489, San Diego, California, USA. Association for Computational Linguistics.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 207–212, Berlin, Germany. Association for Computational Linguistics.

CLARK at SemEval-2019 Task 3: Exploring the Role of Context to Identify Emotion in a Short Conversation

Joseph R. Cummings
Northwestern University
jcummings@u.northwestern.edu

Jason R. Wilson
Northwestern University
jrw@northwestern.edu

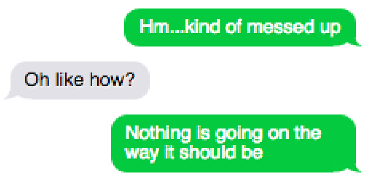
Abstract

With text lacking valuable information available in other modalities, *context* may provide useful information to better detect emotions. In this paper, we do a systematic exploration of the role of context in recognizing emotion in a conversation. We use a Naïve Bayes model to show that inferring the *mood* of the conversation before classifying individual utterances leads to better performance. Additionally, we find that using context while training the model significantly decreases performance. Our approach has the additional benefit that its performance rivals a baseline LSTM model while requiring fewer resources.

1 Introduction

Recognizing affect (emotional content) in text has been an ongoing research challenge for roughly 20 years. While earlier work focused on larger bodies of text, like movie reviews for sentiment analysis (Pang et al., 2002) or classifying mood in blog posts (Mishne et al., 2005), more recent work has looked at small bodies of text, particularly text from social media. With smaller bodies of text inherently having less information, current efforts are investigating how context may supplement the information. However, it is not yet clear how best to incorporate context. To this end, we explore how mood and emotion from previous messages may be used to better recognize emotions.

Mood and emotion are generally regarded as two types of affect. Emotions are reactions and have a limited duration (Ortony et al., 1990; Schwarz and Clore, 2006). While emotions are dynamic and constantly changing, mood reflects a more persistent affect that can influence cognitive processes (Busemeyer et al., 2007), including how people recognize emotions (Schmid and Mast, 2010). For this work, we view mood as the affect present in the whole conversation and emotion as what is expressed in a given turn.



Hm...kind of messed up

Oh like how?

Nothing is going on the way it should be

Figure 1: Example of a three turn conversation

Our goal is to take a short, online conversation (see Figure 1) and categorize the last utterance as happy, sad, angry, or others. In this paper, we present our model Conversational Lexical Affect Recognition Kit (CLARK), which is the result of a systematic exploration into how context may be used during the training and classification phases of a model to improve emotion recognition. To assess context we infer the mood of the conversation and the emotions of previous utterances. Although context would seem to be useful, providing additional information, we find that is only beneficial during classification. Conversely, including context while training the model leads to significantly degraded performance.

2 Related Work

There are several approaches to recognizing affect in a body of text. Many have used classification methods on a large body of text, such as movie reviews (Pang et al., 2002), blogs (Mishne et al., 2005; Mihalcea and Liu, 2006), and fairy tales (Alm et al., 2005; Mohammad, 2011), using techniques like SVMs (Pang et al., 2002; Mishne et al., 2005), Naïve Bayes (Mihalcea and Liu, 2006), HMMs (Ho and Cao, 2012), and Deep Learning (Zhang et al., 2018).

More recently, the rise of instant messaging and social media has led to greater interest in recognizing emotion in a smaller body of text. While lexicon based approaches were initially used for detecting emotions in smaller bodies of text (Thelwall et al., 2010; Staiano and Guerini, 2014), Deep Learning models dominate the recent work (Abdul-Mageed and Ungar, 2017; Chatterjee et al., 2019a).

Our approach is a blend of using a larger and smaller body of text. For the larger body, we detect the mood in a whole conversation. Additionally, we consider a smaller body of text, a single message in a conversation, and detect the emotion in that message. In contrast to many recent approaches using Deep Learning techniques, we use a Naïve Bayes model that requires less data and is trained faster while exhibiting no noticeable degradation in performance in comparison to a baseline SS-LSTM model.

3 Model

We model the task of detecting emotions as a multi-class classification problem. Given a user utterance, the model outputs probabilities of it belonging to the four output classes: happy, sad, angry, or others. Our approach uses CLARK, which at its base level, utilizes a Naïve Bayes model (McCallum and Nigam, 1998) with prior probabilities, which we take to be the frequency of tokens per class. To explore the role of context, we examine several variants of training and classification, detailed later. Keeping the feature set small, we use only unigram and bigrams. We also remove stop words and the following set of punctuation: period, dash, underscore, ampersand, tilde, comma, and backslash. To tokenize the tweets, we utilize Natural Language Toolkit's (NLTK) casual tokenize functionality, which places an emphasis on informal language and is able to pick up emoticons and collections of characters that are semantically equivalent to emoticons, e.g. ':)' is a smiley face.

3.1 Training

The model is trained on three turn conversations from Twitter with the last utterance classified according to the context of the first two utterances via semi-automated techniques (Chatterjee et al., 2019b). 30,160 conversations were provided for training and validation, consisting of 4,243 happy, 5,463 sad, 5,506 angry, and 14,947 others.

We test two variants for training the model, Conv, which we use to infer mood, and only Turn 3 (T3), to calculate feature probabilities given our set of four emotions. Conv consists of all words from the entire conversation, whereas T3 is the third and final utterance.

3.2 Classification

Our classification is a two step chaining process as shown in Algorithm 1. In the first step we find the initial probabilities for each class, denoted by the variable *post*. If we are using mood, denoted by the variable $Mood$, then this distribution is calculated using our model on $Conv$ (see line 7). Otherwise, it is set to the prior probabilities generated from the training (line 9). The resulting probabilities for each class are then used as the priors in step two.

In the second step, we classify the following combinations of individual turns in the conversation: {T3}, {T1, T3}, {T1, T2, T3}. Processing a combination consists of finding the posterior of the first turn and using it as the prior for the next turn and continuing until getting a final posterior, from which we take the highest probability class and return it as the final classification.

Algorithm 1 Classification sequence

```
 1: procedure ClassifyInContext
 2:     $Mood \leftarrow True \vee False$
 3:     $Conv \leftarrow$ words from current conversation
 4:     $default \leftarrow$ prior probabilities of all classes
 5:     $Turns \in \{\{T3\}, \{T1, T3\}, \{T1, T2, T3\}\}$
 6:     if $Mood$ then                                      ▷ Step 1
 7:         $post = $ runModel($Conv, default$)
 8:     else
 9:         $post = default$
10:     end if
11:     for each turn $t$ do                                ▷ Step 2
12:         if $t \in Turns$ then
13:             $post = $ runModel($t, post$)
14:         end if
15:     end for
16:     return argmax($post$)
17: end procedure
```

4 Results

Our results show that inferring mood via Conv in the conversation before recognizing emotion in individual utterances yields improved performance. Furthermore, the best performances focus on the first user, utilizing only the first and third utterances in the second step of classification. We also see that in training the model, the best performance comes from limiting our set to just T3.

Results are organized by analysis on the internal model, followed by a comparison against a baseline Deep Learning model - the one provided by the EmoContext organizers. CLARK is tested with two parameters - classification method and training method. Our best results on the test set yielded a micro F_1 score of 0.5637, roughly equiv-

Classification Method	Acc.	Precision	Recall	F_1
Conv (C)	0.4256	0.2788	0.4170	0.3342
Turn 3 (T3)	0.2651	0.2211	0.4105	0.2874
C, T3	0.5575	0.4073	0.5571	0.4705
T1, T2, T3	0.5308	0.3730	0.5120	0.4316
C, T1, T2, T3	0.5358	0.3823	0.5276	0.4433
T1, T3	0.5369	0.3874	0.5235	0.4431
C, T1, T3	0.5732	0.41915	0.5684	0.4825

Table 1: Results from varying parameter classification method and training on the whole conversation.

Classification Method	Acc.	Precision	Recall	F_1
Conv (C)	0.7125	0.7283	0.5366	0.6178
Turn 3 (T3)	0.7155	0.6366	0.7456	0.6867
C, T3	0.8131	0.7875	0.7766	0.782
T1, T2, T3	0.7997	0.7745	0.7516	0.7629
C, T1, T2, T3	0.7979	0.769	0.7573	0.7631
T1, T3	0.8168	0.7921	0.7778	0.7848
C, T1, T3	**0.8169**	**0.7848**	**0.7892**	**0.7870**

Table 2: Results from varying parameter classification method and training on only T3.

alent to the model from the EmoContext organizers. This score is considerably lower than we got from our evaluations on the training data, possibly attributed to the quality of the labels for the training set versus the test set. We chose not to focus our analysis on the test set because we are not able to do a deep analysis as a result of the data not being readily available at the time. The remaining results we discuss are obtained using a 10-fold cross validation on the training set. Tables 1 and 2 show results from using the 30,160 conversations.

From the difference in results between these tables, it is clear that the biggest improvements in the model comes from training on only T3 as opposed to Conv. The difference in the average micro F_1 score (Rijsbergen, 1979) of training on T3 and the average F_1 score of training on Conv is determined to be statistically significant ($p < 0.005$) using a t-test (Kim, 2015).

Within the classification method, we see that using T1 in coordination with T3 provides F_1 scores at least 14% higher than just classifying with T3. In addition, using Conv consistently provides better performances, albeit close. Thus, the best model is one which utilizes a classification combination of Conv, T1, and T3, with a micro F_1 score of 0.7870.

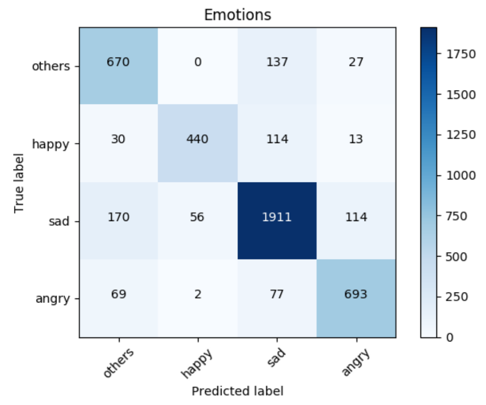


Figure 2: CLARK Confusion Matrix on a 9:1 ratio training to testing split.

As shown in Figure 2, CLARK is incredibly precise in the classification of the sad class and does moderately well in others and angry. However, others and sad are commonly predicted even when not the correct class predicted leading to lower recall scores.

A few concrete examples of these strengths and weaknesses are shown in Table 3. Because CLARK does not place weight on the specific T2 utterance, we see that in no. 2, we miss the positive emotion and misclassify the conversation as others. However, this is largely an anomaly - in fact, we see that T2 usually involves an interrogative or can be associated with a tangential class.

To demonstrate the efficiency of CLARK, we compare it to the baseline SS-LSTM model provided by the SemEval-2019 Task 3 organizers (Chatterjee et al., 2019a) as shown in Table 4. We compare both on time to train and quantity of data needed to produce certain performance. Time to train is normalized to CLARK. The SS-LSTM model performs at 0.6796 when trained with 1 Epoch (used as the very minimum required for a neural model) and takes 26 times longer than CLARK. For 3 Epochs, it performs at 0.7832 and takes 70 times longer than CLARK.

We also examine the effect the size of the training dataset has on the performance of each model, as shown in Figure 3. CLARK vastly outperforms the SS-LSTM models with minimal data. The SS-LSTM (3) model takes the full 30,160 data point

#	T1 (User 1)	T2 (User 2)	T3 (User 1)	True Label	Predicted Label
1	Same to you	:)...hws life going?	Bad	Sad	Sad
2	You are funny	Why, thank you!	So start a fun conversation	Happy	Others
4	Yes	annoys me so much	I really love it	Happy	Happy

Table 3: Results from adjusting classification method and training on only T3.

dataset to achieve an equivalent F_1 score.

Model	Time	Micro F_1
CLARK	1	0.7870
SS-LSTM(1)	26	0.6796
SS-LSTM(3)	70	0.7834

Table 4: Comparison between CLARK and baseline Deep Learning models in terms of normalized time to train and classification performance.

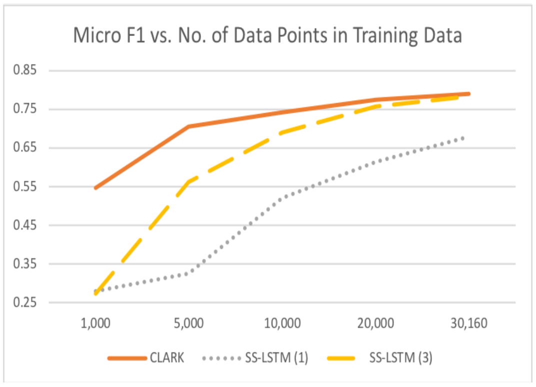


Figure 3: Comparisons of CLARK, SS-LSTM(1), and SS-LSTM(3) at varying amount of training data.

5 Discussion

We investigated a way to model emotion from text in the context of a conversation, instead of a single utterance. In doing so, we analyzed the performance of two different types of models, one based on a Naïve Bayes approach, which we call CLARK, and one on a Deep Learning approach. CLARK trained on T3 and classified using {Conv, T1, T3} leads to the best performance.

One way to utilize context is during training, but our results in experiments with CLARK show that the including more context (i.e., the whole conversation) significantly degrades performance. Training just on T3 produces much better results than training on Conv. This makes sense as T3 is the utterance directly associated with the assigned label and as such, represents the words that we can associate to the label with the highest confidence.

Some notion of "context" is important in determining the overall emotion of a conversation. When classification uses Conv and the final utterance (T3), the model produces the best results, as demonstrated by consistently producing a better F_1 score. This reflects the idea that as humans, mood affects how we judge the emotion a person is currently expressing (Schmid and Mast, 2010).

Our approach to incorporating context is fundamentally different from the approach taken in the baseline model. The SS-LSTM is more similar to a training method using all three utterances and classification method using {T1, T2, T3}. It also takes exponentially longer to train than CLARK and produces roughly equivalent performance, when examining the full dataset. Any attempt to speed up the model by using less training data would be in vain as shown in Figure 3. In cases where efficiency is paramount, the Deep Learning approach is lacking because of these requirements. Being able to produce good results with less training data can be a valuable asset.

Many of this work's limitations come from the data and the way the data was processed. The set of 30,160 three turn conversations is not balanced - there is far more in the others class than the rest. Because Naïve Bayes is a probabilistic model, it will prefer the others class. A solution could be to utilize a Complement Naïve Bayes, which estimates parameters from data using complement classes (Rennie et al., 2003). In addition, the data was labelled using a semi-automatic technique. Human subjects labeled a small subset of tweets, and key word embeddings were then extrapolated to label the rest of the conversations. This method leaves a lot of room for error and even suggests the function our model is trying to learn is this labelling mechanism. In future work, we will use only data labelled by human subjects.

6 Conclusion

Context plays an important role in recognizing emotions, but blindly including context can actually make recognizing emotions more difficult. As a response to the SemEval-2019 Task 3 challenge, we performed a systematic exploration of how to use context in classifying emotions in a short conversation. The resulting model, CLARK, performs best when trained on just the third turn of conversations (no context) and then classification uses Conv to infer mood and emotions from previous turns (with context). The relatively simple Naïve Bayes model, which performs on par with a baseline LSTM model while requiring less data and time to train, demonstrates one successful approach to using context that is usable in resource-constrained scenarios. Furthermore, we believe that while our results are demonstrated using a Naive Bayes model, our approach to using context only when classifying has the potential of being applicable to other classification approaches.

References

Muhammad Abdul-Mageed and Lyle Ungar. 2017. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 718–728.

Cecilia Ovesdotter Alm, Dan Roth, and Richard Sproat. 2005. Emotions from text: machine learning for text-based emotion prediction. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 579–586. Association for Computational Linguistics.

Jerome R. Busemeyer, Eric Dimperio, and Ryan K. Jessup. 2007. Integrating emotional processes into decision-making models. In *Integrated models of cognitive systems.*

Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. 2019a. Understanding emotions in text using deep learning and big data. *Computers in Human Behavior*, 93:309–317.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019b. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Dung T. Ho and Tru H. Cao. 2012. A high-order hidden markov model for emotion detection from textual data. In *Knowledge Management and Acquisition for Intelligent Systems*, pages 94–105, Berlin, Heidelberg. Springer Berlin Heidelberg.

Tae Kyun Kim. 2015. T test as a parametric statistic. In *Korean journal of anesthesiology*.

A. McCallum and K. Nigam. 1998. A comparison of event models for naive bayes text classification. In *Proceedings in Workshop on Learning for Text Categorization*, AAAI98, pages 41–48.

Rada Mihalcea and Hugo Liu. 2006. A corpus-based approach to finding happiness. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, pages 139–144.

Gilad Mishne et al. 2005. Experiments with mood classification in blog posts. In *Proceedings of ACM SIGIR 2005 workshop on stylistic analysis of text for information access*, volume 19, pages 321–327.

Saif Mohammad. 2011. From once upon a time to happily ever after: Tracking emotions in novels and fairy tales. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 105–114. Association for Computational Linguistics.

Andrew Ortony, Gerald L Clore, and Allan Collins. 1990. *The cognitive structure of emotions*. Cambridge university press.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger. 2003. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pages 616–623. AAAI Press.

C. J. Van Rijsbergen. 1979. *Information Retrieval*, 2nd edition. Butterworth-Heinemann, Newton, MA, USA.

Petra Claudia Schmid and Marianne Schmid Mast. 2010. Mood effects on emotion recognition. *Motivation and Emotion*, 34(3):288–292.

Norbert Schwarz and Gerald L. Clore. 2006. Feelings and phenomenal experiences. In A. Kruglansk and E. T. Higgins, editors, *Social psychology: Handbook of basic principles*, 2nd edition, pages 385–407. Guilford, New York.

Jacopo Staiano and Marco Guerini. 2014. Depechemood: a lexicon for emotion analysis from crowd-annotated news. *arXiv preprint arXiv:1405.1605*.

Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 61(12):2544–2558.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253.

CLP at SemEval-2019 Task 3: Multi-Encoder in Hierarchical Attention Networks for Contextual Emotion Detection

Changjie Li
Artificial Intelligence Group
Sogou
Beijing, China, 100000
lichangjie0308@gmail.com

Yun Xing
Artificial Intelligence Lab
Lenovo Research, Lenovo
Beijing, China, 100000
xingyun44@hotmail.com

Abstract

In this paper, we describe the participation of team ”CLP” in SemEval-2019 Task 3 “Contextual Emotion Detection in Text” that aims to classify emotion of user utterance in textual conversation. The submitted system is a deep learning architecture based on Hierarchical Attention Networks (HAN) and Embedding from Language Model (ELMo). The core of the architecture contains two representation layers. The first one combines the outputs of ELMo, hand-craft features and Bidirectional Long Short-Term Memory with Attention (Bi-LSTM-Attention) to represent user utterance. The second layer use a Bi-LSTM-Attention encoder to represent the conversation. Our system achieved F1 score of 0.7524 which outperformed the baseline model of the organizers by 0.1656.

1 Introduction

Emotion detection has been widely researched in psychology, sociology and computer science. Being able to recognize the emotion of text is of vital importance in the human-computer interaction (Cowie et al., 2001). However, detecting emotion in text is generally considered very challenging in absence of facial expression or voice modulation. In domain of natural language processing, emotion detection is a task of associating words, phrases or documents with emotions using psychological models (Duppada et al., 2018). Traditional rule-based approaches (Balahur et al., 2011; Chaumartin, 2007) and machine learning approaches (Alm et al., 2005; Balabantaray et al., 2012) rely on extracting word-level features to classify emotion. These methods suffer from low recall as many texts do not contain emotion words. To tackle the problem, recent deep learning approaches (Mundra et al., 2017) take the advantage of Word2Vec representation (Mikolov et al., 2013a) to extract semantic features and achieve remarkable performance. However, limited researches have been done in classifying textual conversation emotions, which is further compounded by difficulty in the context understanding.

Task 3 “Contextual Emotion Detection in Text” in SemEval-2019 aims to find better solutions for those difficulties in contextual emotion detection (Chatterjee et al., 2019). The task considers textual emotion classification on four-point scale(Happy, Sad, Angry along with an Others category). It classifies emotion of user utterance along with 2 turns of context in conversation.

This paper describes the components and results of our emotion recognition system. The proposed system is a deep learning model based on HAN, which combines multiple encoding methods including ELMo, hand-craft features and Bi-LSTM-Attention encoder. Our system yields a micro-averaged F1 score of 0.7524 on test-set of Task 3 of SemEval 2019.

2 System Description

Figure 1 provides a overall architecture of our approach, which consists of three components: (1) preprocessing, where we use a specially designed text processing method to prepare inputs for our neural network, (2) utterance encoder, where we use ELMo, hand-craft features and Bi-LSTM-Attention encoder to represent user utterance, (3) conversation encoder, where we use a Bi-LSTM-Attention layer to represent the conversation.

2.1 Preprocessing

Twitter limits that a tweet should not exceed 140 characters, which makes users use informal ways to express themselves. Emotion detection for these kinds of tweets is very challenging. To ensure effective feature extraction, we use Ekphrasis (Baziotis et al., 2017) to normalize the utterance.

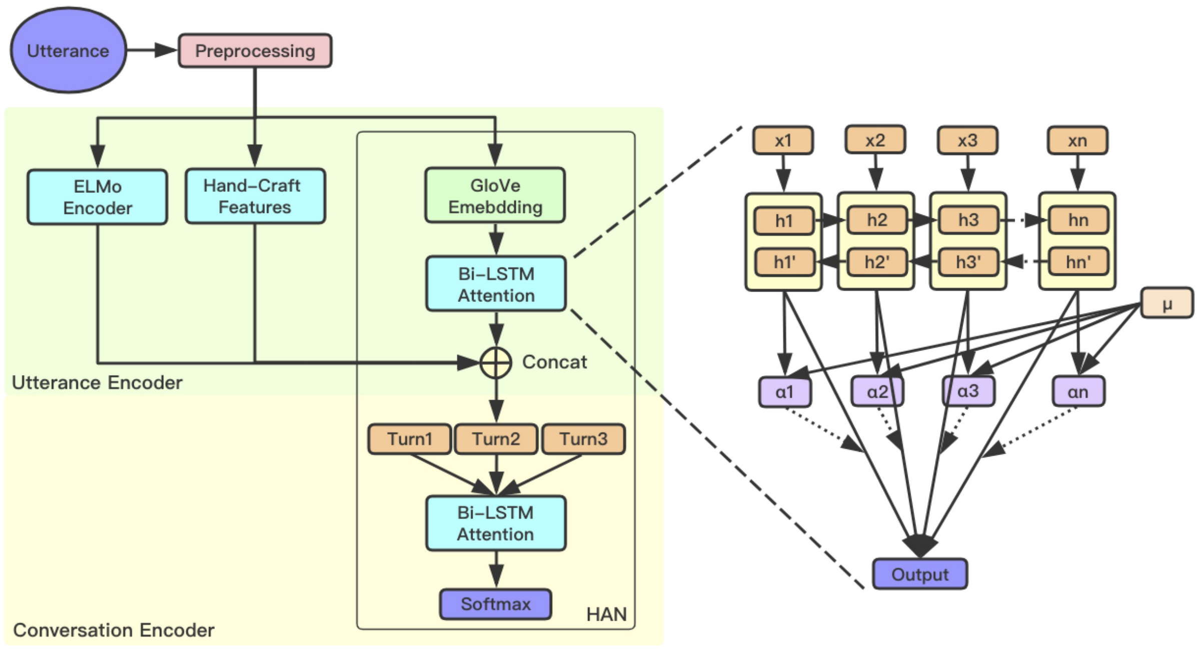


Figure 1: Overall architecture of our approach.

Ekphrasis contains a text processing pipeline that is specially designed for social network texts. The following steps are applied to utterances and lexicons in corpus:

1. **Tokenization.** We use the tokenizer in Ekphrasis to split utterance into word tokens and extract text emoticons from raw texts. The tokenizer is effective in splitting compounded words that are commonly used in Twitter. For example, the output of "#ifeelsad" will be "# i feel sad".

2. **Normalization.** We use regex regressions to detect and normalize categories, such as url/email/money/time/date. These categories are not sensitive features in the task.

3. **Annotation.** We annotate all uppercase words, repeated words and elongated words with corresponding tags, e.g. "hellooooooo" to "hello <elong>"; "yesyesyes" to "yes <repeated>"; "HELLO" to "hello <allcaps>". These informal words are vital features for prediction because they are rich in emotion.

4. **Spelling Correction.** We manually build a dictionary for out-of-vocabulary words (not in pre-trained word vectors) based on the provided datasets. 921 words are collected and corrected. The spelling correction reduces percentage of unknown words from 18% to 12%.

5. **Emoji and Emoticon Normalization.** We normalize emojis/emoticons because some of them have the same meaning. For example, "<3" and "<<33" both indicate heart, while ":(((" and ":(" both represent unhappy.

6. **Lowercase.** All characters in user utterance are converted to lowercase.

2.2 ELMo

ELMo (Peters et al., 2018) is an off-the-shelf pretrained language model that produces deep contextualized word representation, which captures both syntax and semantic information. ELMo can be easily integrated into existing model and usually leads to performance improvement. For most state-of-the-art Natural Language Processing (NLP) tasks, pre-trained word representation is a key component (Mikolov et al., 2013b; Pennington et al., 2014). We assume that different word representations allow the model to benefit from diversified information, so we make ELMo a part of our utterance encoder. Specifically, to generate ELMo representation, we use pre-trained model provided by TensorFlow Hub [1], which outputs a mean-pooling vector of all contextualized

[1]https://tfhub.dev/google/elmo/2.

Dataset	Others	Happy	Sad	Angry	Total
original training	14948	4243	5463	5506	30160
cleaned training	14865	4231	5447	5476	30019
cleaned + augmented training	20351	14566	11240	8319	54476

Table 1: Emotion Distribution of Datasets.

word representations with 1024 dimensions in our model.

2.3 Hand-craft Features

Hand-craft features represent prior knowledge. We extract hand-craft features related to emoji and emoticon because they are frequently used as emotion indicators in Twitter and vital to textual emotion detection. We create a list that contains 300 emojis and emoticons based on this corpus. With the list, we build a Term Frequency–Inverse Document Frequency (TF-IDF) vectorizer. Finally, we convert the utterance to a 300 dimensions vector.

2.4 HAN

HAN (Yang et al., 2016) is designed to capture hierarchical structure in document. Conversation has the same hierarchical structure (words form sentence, sentences form conversation) as document, so we use HAN as the main structure of our system. Our HAN structure has two layers: utterance encoder and context encoder.

Utterance Encoder. We use pre-trained word vectors of GloVe (Pennington et al., 2014) for Twitter as our word embedding. The word embedding is put into a 1-layer Bi-LSTM followed by an attention layer (Vaswani et al., 2017). Figure 1 gives the architecture of Bi-LSTM-Attention. The Bi-LSTM summarizes utterance from both directions and incorporates the contextual information, while the attention mechanism extracts word importance. After the Bi-LSTM-Attention, we combine the output of Bi-LSTM-Attention, ELMo and hand-craft feature vector to represent user utterance.

Conversation Encoder. Given the utterance representation of each turn, we get the conversation representation in a similar way. We use another Bi-LSTM layer to summarize the contextual information in conversation, and we apply attention mechanism to capture the importance of each turn. The output vector of the conversation encoder is a high level representation of the conversation and can be used as features for classification, which is a final softmax layer that predict the emotion.

3 Experiments and Evaluation

3.1 Data Preparation

The organizers provide 30160 conversations for training, 2755 for development and 5509 for test. Before training, we remove conversations that might not be correctly labeled. Then we create more datasets by data augmentation.

Data Cleaning. We firstly train our models with five-fold cross validation. 500 false positive data points with high confidence are picked out. Among them, we manually filter and delete 141 wrong labeled conversations.

Data Augmentation. Data augmentation (DA) is frequently used in Computer Vision (Fawzi et al., 2016). However, this method is less powerful in NLP because NLP data is discrete. Even small perturbations may change the meaning of a whole sentence. In this task, we assume that the positions of emojis and emoticons do not influence the emotion of sentences, so DA can be considered reliable. Our DA includes two steps, 1) all emojis and emoticons in an utterance are extracted by using Ekphrasis, 2) we relocate the emojis and emoticons to the start and the end of the utterance, thus we create 2 additional utterances (not applied for utterances that contain emojis/emoticons only, or utterances begin or end with emojis/emoticons). In total, we get at most 3 utterances for each turn, which means 27 conversations for three turns.

Table 1 describes the emotion distribution of original, data cleaned, data cleaned and augmented training datasets. The proportion of each class in original training dataset is around 4:1:1:1 (Others:Happy:Sad:Angry) and it remains the same after data cleaning. However, DA changes the proportion to around 5:4:3:2 because Twitter users are more likely to use emojis/emoticons when they post happy, sad and angry tweets. In total, 24457 additional data points are created and the distribution of Angry, Sad and Happy classes is improved.

Model	$\mathbf{F1}_{Angry}$	$\mathbf{F1}_{Happy}$	$\mathbf{F1}_{Sad}$	$\mathbf{F1}_{Micro}$
Baseline of Organizers	0.5945	0.5461	0.6149	0.5868
HAN	0.6585	0.6716	0.7667	0.6935
HAN+ELMo	0.6922	0.6973	0.7462	0.7102
HAN+ELMo+HCF	0.7062	0.6997	0.7575	0.7199
HAN+ELMo+HCF+Preprocessing	0.7552	0.6935	0.7959	0.7459
HAN+ELMo+HCF+Preprocessing+DA	**0.7607**	**0.7013**	**0.7961**	**0.7524**

Table 2: Class-wise and micro-averaged F1 scores for models. Our best result comes from HAN + ELMo + Hand-craft Feature (HCF) + Preprocessing + DA.

3.2 Hyper-parameters

We minimize the cross-entropy loss function by using back-propagation with Adam (Kingma and Ba, 2015) and mini-batches of size 64. In order to optimize our results, we introduce class weights in loss function to reduce the impact of the unbalanced training set. The value of class weights is set based on the distribution of classes. The configuration of hyper-parameters includes as follows: the word embedding size is 200; the dimension of hidden layer size in LSTM is 200; the max length of the utterance in each turn is set 25, as nearly 99% of the utterances have less than 25 word tokens; the dropout rate is 0.2 to prevent over-fitting; the learning rate is 10e-3 and the learning rate decay is 10e-5 for each update.

3.3 Results and Discussion

Table 2 presents the results of different approaches. The organizers provide a baseline model with 0.5878 F1 score. Our best model achieves F1 score of 0.7524 which outperforms the baseline by 0.1656. We present 5 models to show how different components (Preprocessing, ELMo, Hand-craft features and DA) affect the performance. The results indicate that HAN is well performed in this task, which alone increases F1 score from 0.5878 to 0.6935. With ELMo encoder and hand-craft features, the performance improves by 0.0264 and continues to rise to 0.7449 if we apply preprocessing to the utterances. DA improves the F1 score of Angry, Sad and Happy, suggesting that more data points of these three classes are beneficial for the task.

Attention weight in HAN reflects how utterances contribute to emotion classification. Therefore, we calculate the average attention weight of three turns in conversation of test-set. Figure 2 shows that the third turn contributes the most to emotion detection, complying with the objective of the task which is to predict third turn emotion. We also find that the weight of first turn is higher than the second, which can be explained by the fact that the first turn and the third turn come from the same user.

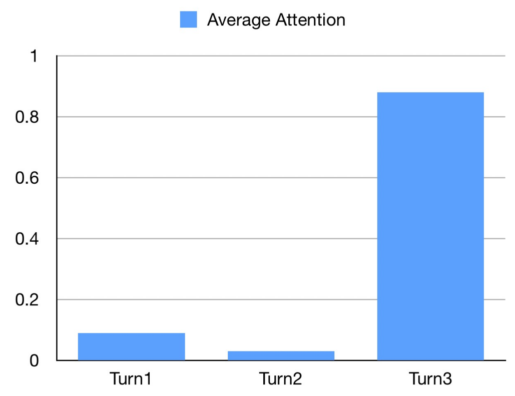


Figure 2: Attention weight of three turns.

4 Conclusion

In this paper we describe our solution to SemEval 2019 Task 3. To classify contextual emotion in a conversation, we propose a HAN based deep learning model that combines multiple encoding methods including ELMo, hand-craft features and Bi-LSTM-Attention encoder. We also build a preprocessing method to improve inputs quality and we apply data augmentation to create more data points. With all these components, our system achieves micro-averaged F1 score of 0.7524 and ranks 17th out of 165 teams on Task 3 leaderboard.

References

Cecilia Alm, Dan Roth, and Richard Sproat. 2005. Emotions from text: Machine learning for text-based emotion prediction.

R C Balabantaray, Iiit Bhubaneswar, Mudasir Mohammad, and Nibha Sharma. 2012. N.: Multi-class twitter emotion classification: A new approach. *International Journal of Applied Information Systems*, pages 48–53.

Alexandra Balahur, Jesús M. Hermida, and Andrés Montoyo. 2011. Detecting implicit expressions of sentiment in text based on commonsense knowledge. In *Proceedings of the 2Nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, WASSA '11, pages 53–60, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada. Association for Computational Linguistics.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

François-Régis Chaumartin. 2007. Upar7: A knowledge-based system for headline sentiment tagging. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 422–425, Prague, Czech Republic. Association for Computational Linguistics.

Roddy Cowie, Ellen Douglas-Cowie, Nicolas Tsapatsoulis, George Votsis, Stefanos Kollias, Winfried Fellenz, and J.G. Taylor. 2001. Emotion recognition in human-computer interaction. *Signal Processing Magazine, IEEE*, 18:32 – 80.

Venkatesh Duppada, Royal Jain, and Sushant Hiray. 2018. Seernet at semeval-2018 task 1: Domain adaptation for affect in tweets. pages 18–23.

A. Fawzi, H. Samulowitz, D. Turaga, and P. Frossard. 2016. Adaptive data augmentation for image classification. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3688–3692.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Shreshtha Mundra, Anirban Sen, Manjira Sinha, Sandya Mannarswamy, Sandipan Dandapat, and Shourya Roy. 2017. Fine-grained emotion detection in contact center chat utterances. pages 337–349.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

CoAStaL at SemEval-2019 Task 3: Affect Classification in Dialogue using Attentive BiLSTMs

Ana Valeria González[*] and **Victor Petrén Bach Hansen**[*] and **Joachim Bingel** and **Isabelle Augenstein** and **Anders Søgaard**
University of Copenhagen, Dept. of Computer Science
Copenhagen, Denmark
ana|victor.petren|bingel|augenstein|soegaard@di.ku.dk

Abstract

This work describes the system presented by the CoAStaL Natural Language Processing group at University of Copenhagen. The main system we present uses the same attention mechanism presented in (Yang et al., 2016). Our overall model architecture is also inspired by their hierarchical classification model and adapted to deal with classification in dialogue by encoding information at the turn level. We use different encodings for each turn to create a more expressive representation of dialogue context which is then fed into our classifier. We also define a custom preprocessing step in order to deal with language commonly used in interactions across many social media outlets. Our proposed system achieves a micro F1 score of 0.7340 on the test set and shows significant gains in performance compared to a system using dialogue level encoding.

1 Introduction

Recognizing emotion is crucial to human-human communication and has for a long time been a goal in human-machine interaction. Although there has been growing interest in emotion detection across many fields (Liscombe et al., 2005; Agrafioti et al., 2012; Craggs and Wood, 2004), much of the work has focused on developing empathetic systems using multimodal approaches i.e. speech and gestures as well as text (Hazarika et al., 2018). Approaching emotion detection as a multimodal problem certainly makes sense, as face-face human communication involves many modalities, however, this fails to consider all the communication that is increasingly happening solely via chat, or written means. Detecting emotion in textual dialogue without the other modalities, such as work done by Gupta et al., can allow us to improve a number of applications dealing with social media interactions, opinion mining, and customer interactions, unfortunately, this is a great challenge that has remained largely unexplored. SemEval 2019 Task 3 attempts to encourage research in this direction. Given a user utterance and the previous two turns of context, the task consists in classifying the user utterance according to one of four emotion classes: happy, sad, angry or other. For a full description of the task see (Chatterjee et al., 2019). In this paper, we describe our turn-level attention model used to tackle this task, specifically using the attention mechanism presented in (Yang et al., 2016). Our model encodes turns in a conversation separately using an Attentive Bidirectional LSTM encoder. In the model presented in the shared task, the turn encoders do not share parameters, achieving a micro F1 score of 0.7340. The code for all experiments presented here is available. [1]

2 Related Work

Due to its many potential applications across many fields, detection of speaker emotional state in spoken dialogue systems has been studied extensively. Early studies showed that the use of prosody as well as speaking style features leads to increases in accuracy for emotion prediction (Ang et al., 2002; Devillers et al., 2002). Researchers have also shown that using domain specific features as well as speech signals can improve performance (Ai et al., 2006). Furthermore, there is plenty of work that tries to improve detection of affect in text using multiple modalities such as video or an embodied conversational agent (Alonso-Martin et al., 2013; Dmello and Graesser, 2010), however, detection of emotion solely based on text conversation data has not seen the same breakthroughs.

[*] Authors contributed equally

[1] https://github.com/coastalcph/emocontext

Very recently, work has started to emerge dealing with emotion detection in textual dialogue only. Majumder et al. introduced an attentive RNN model that treats each party of a conversation independently in order to provide specific representations for speaker and listener at a given point in a conversation. This model assumes that in a dialogue, the emotion of speaker A will be influenced by the utterances expressed by speaker B. Using this approach, the model achieves state of the art performance in two affect datasets (Schuller et al., 2011; Busso et al., 2008)). The model presented by us falls in this line of research, as we also attempt to exploit turn level information independently, however, our work differs in the fact that we create representations for each turn as opposed to creating representations for each speaker.

3 Data Preprocessing

Due to the casual text language used in many of the dialogues in the dataset, properly preprocessing these is an essential part of the classification process. The preprocessing pipeline consists of multiple steps that we describe in more depth below.

Text Normalization We use a custom normalization function which takes commonly used contractions in social media and maps them to a normalized version by unpacking them i.e. *idk* $\rightarrow$ *i don't know* and *plz* $\rightarrow$ *please*.

Spell Correction We normalize elongated words and use a spell correction tool which replaces misspelled words with the most probable candidate based on 2 corpora (Wikipedia and Twitter) (Baziotis et al., 2017a).

Tokenization We employ a tokenizer which emphasizes expressions and words typically used in social media. These include: 1) censored words, 2) words with emphasis, 3) elongated words, 4) splitting emoticons etc. All words are also lowercased when tokenized.

Emoji Descriptions As the dialogues contain a wide variety of emojis, which can contain a great deal of information about a users emotions (Felbo et al., 2017), we replace the emojis found in the utterances with their textual description. We used the emoji descriptions utilized for training Emoji2Vec (Eisner et al., 2016) which can be found in the Unicode emoji standard [2].

For most of the preprocessing steps described above, we relied on the Ekphrasis[3] text processing tool (Baziotis et al., 2017b).

4 Model Description

This section describes our conversational sentiment classification model as was used in the EmoContext shared task. Our architecture is illustrated in figure 1.

Embedding Layer We initiate the embedding layer with an embedding matrix computed using pretrained GloVe embeddings trained on 2 Billion tweets[4]. We do not finetune the weights during training.

Turn Encoder We use bidirectional LSTMs to encode a single turn in the conversation. Given a turn T_k made up of N_k words i.e. $T_k = (w_{1_k}, w_{2_k}, ..., w_{N_k})$, the representation of a given word w_{N_k} consists of the concatenation of the forward hidden state $\overrightarrow{h}_{N_k}$ and backward hidden state $\overleftarrow{h}_{N_k}$, i.e. $h_{N_k} = [\overleftarrow{h}_{N_k}, \overrightarrow{h}_{N_k}]$. This bidirectional representation is then fed through a batch normalization layer and then into the attention layer. The different turn encoders do not share their weights.

Turn Attention We use the attention mechanism introduced by (Yang et al., 2016) in order to extract important words in a single turn. The representation h_{N_k} is fed into a one-layer MLP to obtain the representation u_{N_k}. The similarity between u_{N_k} and a randomly initialized word context vector u_c is computed using the dot product and then the normalized weights α_{N_k} are obtained through a softmax function:

$$\alpha_{N_k} = \frac{exp(u_{N_k} \cdot u_c)}{\sum_k exp(u_{N_k} \cdot u_c)}$$

The final turn representation T_k is the weighted sum of the word vectors based on α_{N_k}.

$$T_k = \sum_k \alpha_{N_k} h_{N_k}$$

[2]http://www.unicode.org/emoji/charts/full-emoji-list.html

[3]https://github.com/cbaziotis/ekphrasis

[4]https://nlp.stanford.edu/projects/glove/

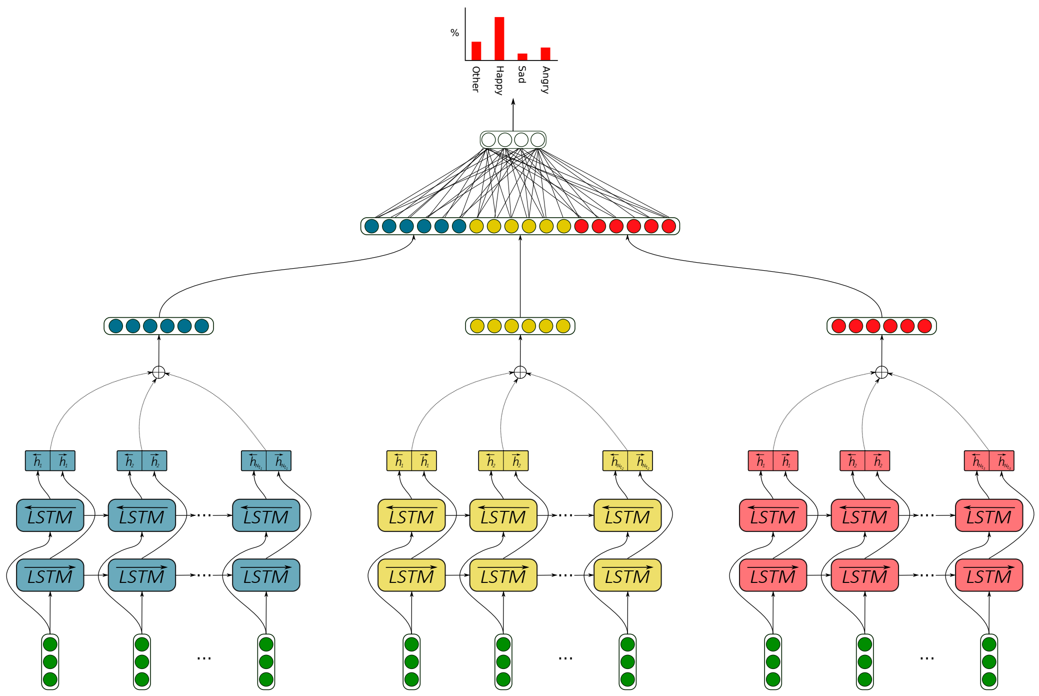


Figure 1: Our proposed model architecture. The turns are preprocessed and embedded using pretrain GloVe embeddings (trained on Twitter data) and fed into their respective BiLSTMs, which are attended over, and combined into a dialogue representation that is classified into one of the 4 classes.

Dialogue Representation For each turn of the conversation we use separate turn encoders and turn attention mechanisms and concatenate the final representations. So for a dialogue of k turns, we end up with a dialogue vector $D = [T_1, T_2, T_3, ..., T_k]$. This representation was chosen as the dialogue length was fixed to 3, but in a variable turn number setting, an LSTM could be utilized to create the final dialogue representation.

Emotion Classification The representation of dialogue D is fed into a softmax layer in order to estimate the probability distribution over the four possible emotion classes.

5 Baselines

Dialogue-level LSTM The main baseline model we compare performance to is the one provided by the task organizers. The system consists of one LSTM encoder, which encodes all turns in the conversation in the same sequence, separated by an end-of-turn token. We show the performance of the baseline given the provided preprocessing script by the organizers. In addition, we include the results of the baseline model using our custom preprocessor.

Dialogue-level TF-IDF with no added features For comparison with the dialogue-level LSTM, we include the performance of a SVM model with stochastic gradient descent. As input we use TF-IDF features computed over all turns. We encode the entire dialogue into a vector of the top 5k features.

Dialogue-level TF-IDF with added features In order to investigate the effect of additional information beyond TF-IDF features, we compute the ratio of words that are 1) elongated and 2) capitalized at the dialogue-level. In addition, we compute the average embeddings of the emojis (Eisner et al., 2016) occurring in the dialogue and concatenate all features with the dialogue level TF-IDF features.

Turn-level TF-IDF with no added features As our main system is a turn encoder, for comparison we also include the performance of an SVM classifier using stochastic gradient descent. using turn level TF-IDF vectors, concatenated into a final dialogue representation.

Turn-level TF-IDF with added features In order to quickly investigate the effect of additional information at the turn level, we compute the same features as mentioned earlier: 1) the ratio of words that are capitalized in a given turn, 2) the ratio of words that are elongated, and 3) the average embeddings of the emojis (Eisner et al., 2016) occurring in a given turn. All features are concatenated.

6 Setup and Results

In addition to our proposed system that is described in Section 4 (BiLSTM-Att) and the baselines in Section 5, we also report results for two other variants of BiLSTM-Att. The first model (BiLSTM-Att-Shared) shares weights between the RNNs, ie. we encode all turns individually with the same BiLSTM, and a model that simply encodes the entire dialogue as a single turn (BiLSTM-Att-Dia). We train the models with BiLSTM hidden state size of 256, a dropout rate between the LSTM layer and attention layer of 0.5 , a batch size of 200, and we use a word embedding dimension of 200. We optimize using the ADADELTA algorithm (Zeiler, 2012) with a learning rate of 1.0, $\rho = 0.95$ and $\varepsilon = 10^{-6}$.

System	Micro F1	Precision	Recall
LSTM	0.5613	0.4743	0.6862
Dialogue-tfidf	0.5528	0.6202	0.4986
Dialogue-tfidf-added	0.6064	0.7127	0.5276
Turn-tfidf	0.5918	0.6394	0.5507
Turn-tfidf-added	0.6955	**0.7632**	0.6388
BiLSTM-Att	**0.7340**	0.7132	0.7560
BiLSTM-Att-Shared	0.7243	0.6715	**0.7861**
BiLSTM-Att-Dia	0.6789	0.6039	0.7752

Table 1: The table shows the results of our models on the EmoContext shared task test set.

Our results are shown in Table 1. From the results we can observe that our proposed attentive turn-level BiLSTM outperforms all baselines, including the task organizers LSTM model, with a Micro F1 score of 0.7340. What is interesting to note is that almost all of our proposed simple SVM baselines also outperforms the baseline LSTM, with even Turn-tfidf-added by a significant margin. In general we see that encoding the dialogue on the turn level achieves better performance than its dialogue level counterparts.

7 Discussion

We saw that in all cases, encoding information at the turn level led to improvements in classifier performance over the dialogue level encoding. This observation is in line with work that exists trying to encode conversational context beyond the single turn or the dialogue level (Majumder et al., 2019; Webb et al., 2005). In addition, in the results shown in Table 1, we can observe that the dialogue level attention LSTM achieves a high recall but low precision. In contrast, the differences in all metrics for our proposed model are much smaller and more balanced. This suggests that without the turn level encoding, the classifier becomes more biased towards a specific class. In Table 2, we show the scores of the individual emotion classes for our turn and dialogue-level models. We can see that across all classes the models have a harder time when it comes to classifying dialogues labeled as Happy, suggesting that the happy conversations might have a tendency to be more neutral in language, resulting in a higher mislabelling rate with the Other class. This becomes more apparent when inspecting the data itself. What is also noteworthy is that the BiLSTM-Att-Shared, which shared a turn encoder between turns, achieves a lower F1 score than BiLSTM-Att, which used separate turn encoders. This could indicate that the different turns carry different weights in the context when it comes to determining the sentiment of the most recent speaker.

Emotion class	Micro F1	Precision	Recall
BiLSTM-Att			
Angry	0.751	0.686	0.829
Happy	0.692	0.721	0.666
Sad	0.757	0.742	0.772
BiLSTM-Att-Dia			
Angry	0.699	0.590	0.859
Happy	0.648	0.601	0.704
Sad	0.687	0.687	0.760

Table 2: F1, precision and recall scores for each of the emotion classes for two of our proposed models, BiLSTM-Att and BiLSTM-Att-Dia

8 Conclusion

Overall, our very straight forward model shows the important effect that encoding turn level information separately has when it comes to classifying dialogues. Using the entire dialogue with an end-of-turn token, we see that the model is not able to capture important features of individual turns that might affect the overall sentiment of the conversation. Our results also shows that, although less sophisticated, simpler and more interpretable models does also give decent results, compared to the LSTM baseline model.

References

Foteini Agrafioti, Dimitris Hatzinakos, and Adam K Anderson. 2012. Ecg pattern analysis for emotion detection. *IEEE Transactions on Affective Computing*, 3(1):102–115.

Hua Ai, Diane J Litman, Kate Forbes-Riley, Mihai Rotaru, Joel Tetreault, and Amruta Purandare. 2006. Using system and user performance features to improve emotion detection in spoken tutoring dialogs. In *Ninth International Conference on Spoken Language Processing*.

Fernando Alonso-Martin, Maria Malfaz, Joao Sequeira, Javier Gorostiza, and Miguel Salichs. 2013. A multimodal emotion detection system during human–robot interaction. *Sensors*, 13(11):15549–15581.

Jeremy Ang, Rajdip Dhillon, Ashley Krupski, Elizabeth Shriberg, and Andreas Stolcke. 2002. Prosody-based automatic detection of annoyance and frustration in human-computer dialog. In *Seventh International Conference on Spoken Language Processing*.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017a. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017b. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada. Association for Computational Linguistics.

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan. 2008. Iemocap: Interactive emotional dyadic motion capture database. *Language resources and evaluation*, 42(4):335.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Richard Craggs and Mary McGee Wood. 2004. A categorical annotation scheme for emotion in the linguistic content of dialogue. In *Tutorial and Research Workshop on Affective Dialogue Systems*, pages 89–100. Springer.

Laurence Devillers, Ioana Vasilescu, and Lori Lamel. 2002. Annotation and detection of emotion in a task-oriented human-human dialog corpus. In *proceedings of ISLE Workshop*.

Sidney K Dmello and Arthur Graesser. 2010. Multimodal semi-automated affect detection from conversational cues, gross body language, and facial features. *User Modeling and User-Adapted Interaction*, 20(2):147–187.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bosnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. *CoRR*, abs/1609.08359.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625. Association for Computational Linguistics.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations. *CoRR*, abs/1707.06996.

Devamanyu Hazarika, Soujanya Poria, Amir Zadeh, Erik Cambria, Louis-Philippe Morency, and Roger Zimmermann. 2018. Conversational memory network for emotion recognition in dyadic dialogue videos. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2122–2132.

Jackson Liscombe, Giuseppe Riccardi, and Dilek Hakkani-Tur. 2005. Using context to improve emotion detection in spoken dialog systems.

Navonil Majumder, Soujanya Poria, Devamanyu Hazarika, Rada Mihalcea, Alexander Gelbukh, and Erik Cambria. 2019. Dialoguernn: An attentive rnn for emotion detection in conversations. In *proceedings of AAAI Conference*.

Björn Schuller, Michel Valstar, Florian Eyben, Gary McKeown, Roddy Cowie, and Maja Pantic. 2011. Avec 2011–the first international audio/visual emotion challenge. In *Affective Computing and Intelligent Interaction*, pages 415–424. Springer.

Nick Webb, Mark Hepple, and Yorick Wilks. 2005. Dialogue act classification based on intra-utterance features. In *Proceedings of the AAAI Workshop on Spoken Language Understanding*, volume 4, page 5. Citeseer.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

ConSSED at SemEval-2019 Task 3: Configurable Semantic and Sentiment Emotion Detector

Rafał Poświata
National Information Processing Institute
al. Niepodległości 188b, 00-608 Warsaw, Poland
rafal.poswiata@opi.org.pl

Abstract

This paper describes our system participating in the SemEval-2019 Task 3: EmoContext: Contextual Emotion Detection in Text. The goal was to for a given textual dialogue, i.e. a user utterance along with two turns of context, identify the emotion of user utterance as one of the emotion classes: Happy, Sad, Angry or Others. Our system: ConSSED is a configurable combination of semantic and sentiment neural models. The official task submission achieved a micro-average F1 score of 75.31 which placed us 16th out of 165 participating systems.

1 Introduction

Emotion detection is crucial in developing a "smart" social (chit-chat) dialogue system (Chen et al., 2018). Like many sentence classification tasks, classifying emotions requires not only understanding of single sentence, but also capturing contextual information from entire conversations. For the competition we were invited to create a system for emotion detection of user utterance from short textual dialogue i.e. a user utterance along with two turns of context (Chatterjee et al., 2019b). The number of emotion classes has been limited to four (Happy, Sad, Angry and Others).

The rest of the paper is organized as follows. Section 2 briefly shows the related work. Section 3 elaborates on our approach. It shows preprocessing step and architecture of our system. Section 4 describes the data set, used word embeddings and hyper-parameters, adopted research methodology and experiments with results. Finally, Section 5 concludes our work.

2 Related Work

Detection of emotions in dialogues can be divided into two types: based only on the text of the dialogue (Chen et al., 2018) and based on many channels (video, speech, motion capture of a face, text transcriptions) (Busso et al., 2008). Regardless of the type, the most common solution is the use of neural networks, in particular variations of Recurrent Neural Networks, such as LSTMs (Hochreiter and Schmidhuber, 1997), BiLSTMs (Schuster and Paliwal, 1997) and GRUs (Cho et al., 2014) or Convolutional Neural Networks (Krizhevsky et al., 2012). Our solution uses LSTMs and BiLSTMs and is based on the ideas from SS-BED system (Chatterjee et al., 2019a).

3 Our Approach

Figure 1 provides an overview of our approach. We wanted to create a system that would benefit from the advantages of semantic and sentiment embeddings (like SS-BED). At the same time, it would be easily configurable both in terms of the selection of parameters/network architecture as well as the change of applied embeddings, both static and dynamic. In the next subsections, we describe in details our approach.

3.1 Preprocessing

For the preprocessing, we adjusted the ekphrasis tool (Baziotis et al., 2017). We use this tool for tokenization and to do the following:

- Normalize URLs, emails, percent/money/time/date expressions and phone numbers.
- Annotate emphasis and censored words and phrases with all capitalized letters.
- Annotate and reduce elongated (e.g. Whaaaat becomes <elongated> What) and repeated words (e.g. !!!!!!!!! becomes <repeated> !).

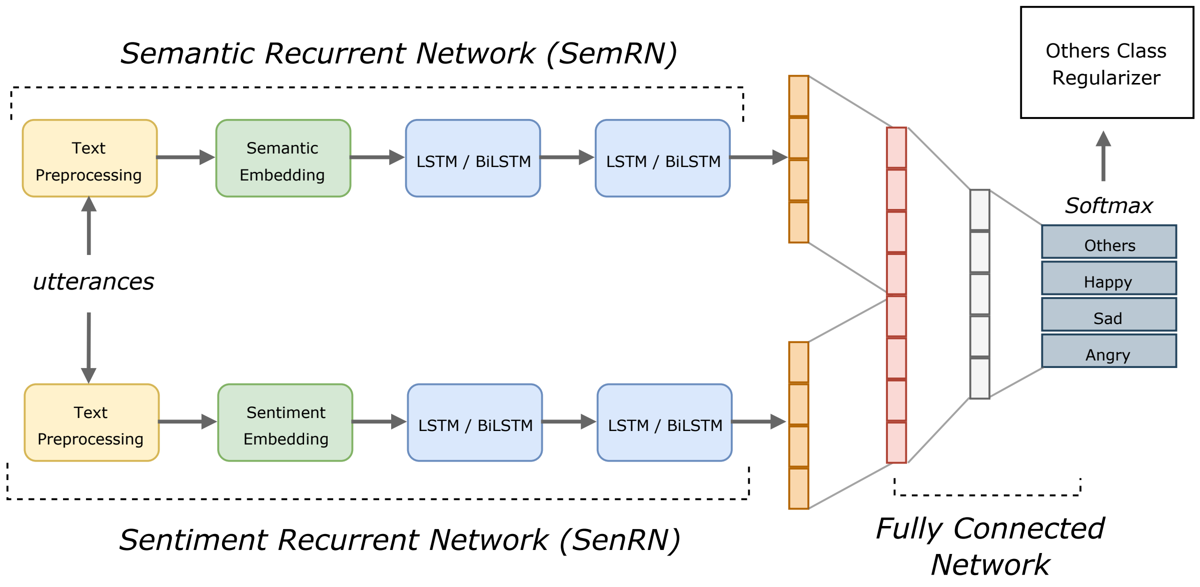


Figure 1: High level architecture of Configurable Semantic and Sentiment Emotion Detector (ConSSED).

- Unpack hashtags (e.g. #GameTime becomes <hashtag> game time </hashtag>) and contractions (e.g. "didn't" becomes "did not").
- Simplify emoticons e.g. :-] is changed to :).

We also prepare and apply dictionaries with common abbreviations and mistakes to reduce vocabulary size and deal with Out of Vocabulary (OOV) issue.

3.2 Model

Our model contains four parts: Semantic Recurrent Network (SemRN), Sentiment Recurrent Network (SenRN), Fully Connected Network and Others Class Regularizer. SemRN and SenRN are independent of each other and have similar architecture: Text Preprocessing, suitable Word Embedding and 2-layer LSTM or bidirectional LSTM (BiLSTM) - which is configurable. Outputs of those two modules are concatenated and become input for Fully Connected Network. This network has one hidden layer and Softmax layer which represents probabilities of classes. The last element of our model is Others Class Regularizer (used only during the prediction on validation/test set).

3.3 Others Class Regularizer

This component was created due to the fact that a real-life distribution is about 4% for each of Happy, Sad and Angry class and the rest is Others class. This component works by grouping records into three sets, depending on whether they are predicted as Happy, Sad or Angry. Next, for all of these sets, it checks if there are more representatives than the assumed percentage of all records. If yes, it increases the probability for Others class by 0.01 (independently in each set) until it reaches the number of representatives lower than the assumed percentage. The assumed percentage value was defined as 5.5% taking into account the validation set.

4 Experiments and Results

4.1 Data

In our work on the system, we used only official data sets made available by the organizers. However, we noticed that there are cases when conversations occur twice, but with different labels. We have removed these records and received sets which are shown in Table 1.

	Number of records
train	29977
validation	2755
test	5509

Table 1: Data sets statistics.

4.2 Word Embeddings

For our experiments, we chose five word embeddings: three semantic and two sentiment. Semantic embeddings are GloVe (Pennington et al., 2014) trained on Twitter data[1], Word2Vec (Mi-

[1]https://nlp.stanford.edu/projects/glove/

Hyper-parameter name	Possible values
SEM_LSTM_DIM	[200, 230, 256, 280, 300, 320]
SEM_FIRST_BIDIRECTIONAL	[False, True]
SEM_SECOND_BIDIRECTIONAL	[False, True]
SEN_LSTM_DIM	[200, 230, 256, 280, 300, 320]
SEN_FIRST_BIDIRECTIONAL	[False, True]
SEN_SECOND_BIDIRECTIONAL	[False, True]
HIDDEN_DIM	[100, 128, 150]
LSTM_DIM	[200, 230, 256, 280, 300, 320]
BATCH_SIZE	[32, 64, 80, 100, 128]
DROPOUT	(0.1, 0.5)
RECURRENT_DROPOUT	(0.1, 0.5)
LEARNING_RATE	(0.001, 0.004)
OTHERS_CLASS_WEIGHT	(1.0, 3.0)

Table 2: The names of hyper-parameters with possible values.

kolov et al., 2013) with ten affective dimensions trained by NTUA-SLP team as part of their solution for SemEval2018 (Baziotis et al., 2018)[2] (we call it NTUA_310) and ELMo (Peters et al., 2018) trained on 1 Billion Word Benchmark[3]. As sentiment embeddings we chose Sentiment-Specific Word Embedding (SSWE) (Tang et al., 2014)[4] and Emo2Vec (Xu et al., 2018)[5].

4.3 Hyper-parameters Search

In order to tune the hyper-parameters of our model, we adopt a Bayesian optimization by using Hyperopt library[6]. The names of hyper-parameters with possible values (list or range) are shown in Table 2. Parameters with SEM prefix apply to the Semantic Recurrent Network, and with SEN prefix to the Sentiment Recurrent Network. LSTM_DIM parameter is for BiLSTM baseline systems. In order to cope with the differences in the distribution of classes in the training set and the validation and test sets, as well as the previously mentioned actual distribution of emotion classes in relation to the Others class, apart from the use of Others Class Regularizer we also used class weight for Others class (OTHERS_CLASS_WEIGHT parameter).

4.4 Methodology

We train all models using the training set and tune the hyper-parameters using the validation set. Due to the time frame of the competition, we limited the search of hyper-parameters to 10 iterations for each model. Then, for the best parameters (found in a limited number of iterations), we once again learned this model with a training and validation set. The final model validation took place on the test set. During all experiments, we used the preprocessing described in section 3.1.

4.5 Experiments

The results of our experiments are shown in Table 3. We have divided them into two stages: validation of the baseline systems and our solution.

For the first stage, we used the 2-layer bidirectional LSTM model (BiLSTM) with all the word embedding presented in section 4.2 and compared this approach to the baseline model prepared by the organizers (Baseline). The model using NTUA_310 embedding (73.34) performed best, compared to the Baseline, we have an improvement of about fifteen percent. The second best model was a solution using ELMo embedding (72.42). From sentiment embeddings the best was Emo2Vec (71.18).

The second stage was focused on the validation of the ConSSED model. In this experiment, we trained six models to verify all possible pairs of semantic embedding-sentiment embedding. The results show that the use of the ConSSED model allows better results than corresponding baseline systems. As we could have guessed from the first stage, the best was a combination of NTUA_310 and Emo2Vec (75.31), which was our official solution during the competition. In parentheses, we presented the results without the use of Others Class Regularizer. As we can see, the use of this component improves the results but only slightly. In addition, after the competition, we have rerun the search for hyper-parameters (this time increasing the number of iterations) for the ConSSED-

[2]https://github.com/cbaziotis/ntua-slp-semeval2018

[3]https://tfhub.dev/google/elmo/2

[4]http://ir.hit.edu.cn/~dytang/

[5]https://github.com/pxuab/emo2vec_wassa_paper

[6]https://hyperopt.github.io/hyperopt/

	Happy F1	**Sad F1**	**Angry F1**	**Avg. F1**
Baseline	54.61	61.49	59.45	58.61
BiLSTM-GloVe	59.62	67.16	73.64	67.39
BiLSTM-ELMo	67.99	74.69	74.35	72.42
BiLSTM-NTUA_310	70.29	77.21	73.07	73.34
BiLSTM-SSWE	66.34	71.54	69.07	68.86
BiLSTM-Emo2Vec	69.48	73.27	70.93	71.18
ConSSED-GloVe-SSWE	68.48 (67.86)	74.91 (69.69)	76.54 (74.00)	73.30 (70.62)
ConSSED-GloVe-Emo2Vec	68.46 (68.46)	77.51 (77.51)	73.21 (71.39)	72.90 (72.18)
ConSSED-ELMo-SSWE	69.27 (69.16)	79.30 (79.30)	74.88 (73.32)	74.27 (73.60)
ConSSED-ELMo-Emo2Vec	71.30 (71.30)	76.05 (76.05)	76.67 (76.50)	74.69 (74.68)
ConSSED-NTUA_310-SSWE	70.69 (70.69)	78.13 (78.13)	75.54 (74.92)	74.66 (74.45)
ConSSED-NTUA_310-Emo2Vec	**69.69 (69.69)**	**78.39 (78.39)**	**77.67 (76.95)**	**75.31 (75.10)**
*ConSSED-NTUA_310-Emo2Vec	72.66 (72.66)	79.60 (79.60)	77.80 (76.83)	76.64 (76.31)

Table 3: Results of our experiments on the test set. The values without the use of Others Class Regularizer are shown in parentheses. Bolded model indicate our official solution in the competition. Experiment with an asterisk was carried out after the end of the competition.

Avg. F1	**Competition Model** **75.31**	**Best Model** **76.64**
SEM_LSTM_DIM	320	320
SEM_FIRST_BIDIRECTIONAL	True	True
SEM_SECOND_BIDIRECTIONAL	False	False
SEN_LSTM_DIM	256	280
SEN_FIRST_BIDIRECTIONAL	True	True
SEN_SECOND_BIDIRECTIONAL	True	True
HIDDEN_DIM	150	150
BATCH_SIZE	100	100
DROPOUT	0.30328	0.34468
RECURRENT_DROPOUT	0.31007	0.29362
LEARNING_RATE	0.00338	0.00333
OTHERS_CLASS_WEIGHT	2.41235	2.63698

Table 4: Comparison between two ConSSED-NTUA_310-Emo2Vec models: official **Competition Model** and **Best Model** trained after the end of the competition.

NTUA_310-Emo2Vec model, which give us a better result than our official competition result (76.64). Hyper-parameters found for ConSSED-NTUA_310-Emo2Vec models and differences between them are shown in Table 4.

4.6 Competition Results

The best result we have obtained on official leaderboard is equal to 75.31 according to micro-averaged F1 score. Our solution is ranked 16th out of 165 participating systems.

5 Conclusion

In this paper, we present Configurable Semantic and Sentiment Emotion Detector (ConSSED) - our system participating in the SemEval-2019 Task 3. ConSSED has achieved good results, and subsequent studies show that it can achieve even better which results from a further search for hyper-parameters. We think that the use of fine-tuned ELMo model (e.g. by Twitter data) would improve the result even more. In addition, we would like to integrate our system with the BERT embedding (Devlin et al., 2018).

For developing our system we used Keras[7] with TensorFlow[8] as backend. We make our source code available at `https://github.com/rafalposwiata/conssed`.

References

Christos Baziotis, Athanasiou Nikolaos, Alexandra Chronopoulou, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, Shrikanth Narayanan, and Alexandros Potamianos. 2018. NTUA-SLP at semeval-2018 task 1: Predicting affective content in tweets with deep attentive rnns and transfer learning. In *Proceedings of The 12th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT, New Orleans, Louisiana, June 5-6, 2018*, pages 245–255.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task

[7]`https://keras.io/`
[8]`https://www.tensorflow.org/`

4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada. Association for Computational Linguistics.

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower Provost, Samuel Kim, Jeannette Chang, Sungbok Lee, and Shrikanth Narayanan. 2008. Iemocap: Interactive emotional dyadic motion capture database. *Language Resources and Evaluation*, 42:335–359.

Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. 2019a. Understanding emotions in text using deep learning and big data. *Computers in Human Behavior*, 93:309–317.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019b. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Sheng-Yeh Chen, Chao-Chun Hsu, Chuan-Chun Kuo, Ting-Hao K. Huang, and Lun-Wei Ku. 2018. Emotionlines: An emotion corpus of multi-party conversations. *CoRR*, abs/1802.08379.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA. Curran Associates Inc.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45:2673–2681.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1555–1565, Baltimore, Maryland. Association for Computational Linguistics.

Peng Xu, Andrea Madotto, Chien-Sheng Wu, Ji Ho Park, and Pascale Fung. 2018. Emo2vec: Learning generalized emotion representation by multi-task training. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 292–298. Association for Computational Linguistics.

CX-ST-RNM at SemEval-2019 Task 3: Fusion of Recurrent Neural Networks Based on Contextualized and Static Word Representations for Contextual Emotion Detection

Michał Perełkiewicz
National Information Processing Institute
al. Niepodległości 188b, Warsaw, Poland
michal.perelkiewicz@opi.org.pl

Abstract

In this paper, I describe a fusion model combining contextualized and static word representations for approaching the EmoContext task in the SemEval 2019 competition. The model is based on two Recurrent Neural Networks, the first one is fed with a state-of-the-art ELMo deep contextualized word representation and the second one is fed with a static Word2Vec embedding augmented with 10-dimensional affective word feature vector. The proposed model is compared with two baseline models based on a static word representation and a contextualized word representation, separately. My approach achieved officially 0.7278 microaveraged F1 score on the test dataset, ranking 47th out of 165 participants.

1 Introduction

The EmoContext task in the Semantic Evaluation 2019 (SemEval 2019) competition focuses on the classification of textual dialogues i.e. a user short conversation with a bot, into ’happy’, ’sad’, ’angry’ and ’others’ sentiment classes. Understanding emotions in textual conversations is a challenging task mainly because of an absence of others expression channels such as voice modulations and facial expressions usually accompanying a people conversation. In textual conversation determining the emotion of a given statement is very dependent on the context of previous statements.

A sentiment detection field has been thoroughly analysed. The first attempts to manage this problem included mainly extracting hand-crafted features and knowledge-based systems (Balahur et al., 2011; Chaumartin, 2007; Joulin et al., 2016).

Neural machine learning approaches, especially Recurrent Neural Networks (RNN) like LSTM and GRU and Convolutional Neural Networks (CNN), are effective tools for detecting sentiment from text and were widely used in this area (Tang et al., 2015; Liu et al., 2016; dos Santos and Gatti, 2014). Except that, one of the approaches employs Hierarchical Attention Networks (HAN) to determining emotions from textual dialogues data (Saxena et al., 2018).

In this work, I propose a deep neural fusion model that combines two Bidirectional LSTM Recurrent Neural Networks for detecting sentiments in textual dialogues. I use static Word2Vev word representation and a contextualized ELMo word representation to create a unified model. This architecture of the model is inspired by the work presented in (Gupta et al., 2017).

The rest of the paper is structured as follows. Section 2 describes the proposed approach and word representations. The experiments and results are presented and discussed in Section 3. Finally, in the last section, the conclusions are presented.

2 Approach

The following section provides details on preprocessing I used to normalize textual data provided by the task organizers, the method to manage unbalanced datasets problem and describes the model architecture and used word embeddings.

2.1 Preprocessing

To clean and normalize textual data, I adapt the *ekphrasis* text processing library[1] with some changes. It was designed with a focus on text from social networks, such as Twitter or Facebook. It provides tools to process text, such as tokenization, word normalization, word segmentation and spell correction, using word statistics from 2 big corpora (English-language Wikipedia and Twitter)(Baziotis et al., 2017).

The *ekphrasis* preprocessing techniques I used includes: Twitter-specific tokenization, omitting

[1] https://github.com/cbaziotis/ekphrasis

special words and phrases (like emails, phone numbers, nicknames, dates and time, URLs), spell correction, words annotations (for uppercased, repeated, hashtagged and elongated words), reducing emoticons variations by replacing emoticons expressing similar emotions to the same form (e.g ':)' and ':-)' emoticons are both mapped to the ⟨happy⟩ mark).

To better normalize texts and suit the tool to EmoContext datasets, I did following modifications in the text processing library:

– adding a new dictionary to expand English contractions (e.g. can't → can not, couldn't've → could not have), normalize slang words (e.g. plz → please) and correct typos (whhat → what).

– extending the emoticons dictionary with emoticons found in the training and the testing corpora to reduce them to the basic form.

– adding new map for emoticons expressing strong emotions by adding the *very* word before them (e.g ':)))))))' → very ⟨happy⟩)

– changing emoticons mapping by adding the prefix 'emo_' to emotion marks[2], like ':)': '⟨emo_happy⟩'.

2.2 Altering the Training Balance

According to information pointed out by the task organizers on the EmoContext web page[3], provided datasets are unbalanced. Training data consists of about 5000 (about 17%) samples each from 'angry', 'sad', 'happy' class, and about 15000 (about 50%) samples from 'others' class, whereas, both the development dataset and the test dataset sets have a real-life distribution, which is about 4% each of 'angry', 'sad', 'happy' class and the rest is 'others' class.

To deal with unbalanced datasets and avoid to bias model towards the 'other' class, I created two derivative training datasets: the binary one, by mapping 'happy', 'sad', 'angry' labels to the one 'sentiment' label and left the 'others' label unchanged. This binary set contains about 50% of dialogues covey some sentiment and about 50% dialogues conveys no sentiment (the original 'other' label). The second derivative dataset contains dialogues labelled one of the following labels: 'happy', 'sad', 'angry'. So, this dataset contains examples originally conveys some sentiment. Classes distributions of these two derivative datasets are more balanced. These derivative datasets are used to learn a two-stage model based on two Recurrent Neural Networks as described in Subsection 2.4.

Furthermore, I extend the training dataset by carrying 1753 examples from the development dataset and left 1000 examples to valid my model.

[2]only used by the contextualized embedding

[3]https://competitions.codalab.org/competitions/19790#learn_the_details-data-set-format

2.3 Word Embeddings

Word embeddings are representations of words as n-dimensional vectors, previously learned on large text corpus. The proposed model is fed both a static word embedding and a contextualized (dynamic) word embedding.

Static Word Embedding Static word embeddings map the same word to the same vector, independently of the word context. The advantages of static word embeddings are easy interpretability and capturing semantic properties of words (embedding vectors for words semantically similar are similar as well). However, they suffer from some problems, for example a meaning conflation deficiency – the inability to discriminate among different meanings of a word.

To embed textual data to a static representation I adapt pretrained 300-dimensional Word2Vec word embedding vector augmented with a 10-dimensional vector of word affective features proposed in (Baziotis et al., 2018). It was trained on the collection of 550 million Twitter messages preprocessed by the *ekphrasis* text processing library. Such very similar preprocessing stage can better suit EmoContext textual data to this pretrained embedding.

Therefore, the static word embedding I use maps every sentence in EmoContext datasets to a 310 dimensional n-length list where n is a number of words in a sentence.

Contextualized Word Embedding As opposed to static embeddings, contextualized word embeddings generate words representation vectors dynamically, depending on the context in which a given word appears. They need the whole sentence to generate words embeddings because of a need to know the context of each word in a sentence.

I employee the Embeddings from Language Models (ELMo) word embedding, where word vector representations are learned functions of the internal states of a deep bidirectional language model (biLM) (Peters et al., 2018). I use official available, pretrained ELMo original model[4] which was learned on the dataset of 5.5 billion tokens consisting of Wikipedia (1.9 billion) and all of the monolingual news crawl data from WMT 2008-2012 (3.6 billion). To vectorize words, I get the state of the last biLM layer built on 1024 neurons, therefore the embedding vectors are the length of 1024 elements. To better suit this embedding to the EmoContext datasets, I fine-tuned the ELMo model by learning pretrained biLM. For this purpose, I used all utterances from the datasets provided by the organizers.

To generate more context-aware representations of words, for each dialogue in the datasets I merged the previous one or two utterances with the second or the third utterance in a dialogue, respectively. Such a context extending allows generating vector representations for two last utterances taking into account the context of the previous utterances in whole dialogue. For a first utterance, I use only the first utterance without any extension.

2.4 Model Architecture

Next, I present in detail the submitted model. My final model is based on the fusion of two deep, two-layer Bidirectional LSTM Neural Networks with Attention Mechanism. First one consumes 310-dimensional vectors and produces a 250-dimensional encoding as a averaged Bidirectional LSTM network states over time. The second one consumes 1024-dimensional vectors and produces a 300-dimensional encoding vector, as in the previous case, as a averaged Bidirectional LSTM network states over time. Then these two encodings are merged and are consumed by a Fully Connected layer with 120 neurons. The activation function of this layer is a softmax function to get probabilities of output classes. The architecture is depicted in Figure 1.

To avoid overfitting during learning the model, I use the following regularization techniques:

– Dropout inputs to BiLSMT networks, for each layer.

– Dropout input to Fully Connected Layer.

[4] https://allennlp.org/elmo

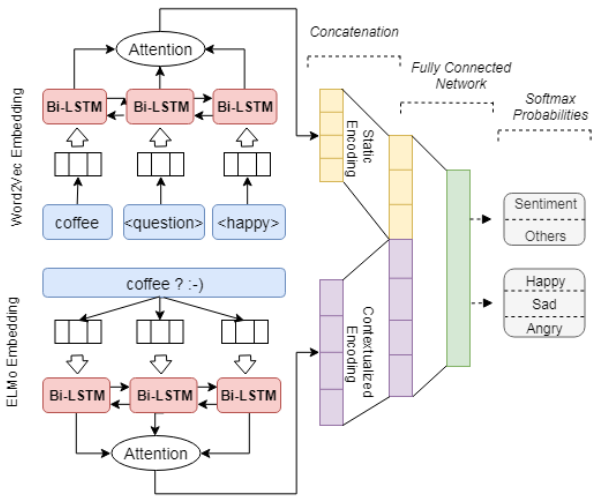


Figure 1: The proposed model architecture.

– Recurrent Dropout to dropout connections between the recurrent units in Recurrent Networks for each layer.

I learnt two models based on described architecture separately using datasets describing in Section 2.2. These two networks are stacked and the prediction process runs as follows:

– the general model (called sent-others network) predicts if a dialogue conveys some sentiment or not. This model predicts one of two labels: ’sentiment’ or ’others’ to input data.

– if the general model has predicted the ’sentiment’ label, the input data is carried forward to the second model (called happy-sad-angry network). The responsibility of this model is to determine the sentiment of an input data and put one of three labels: ’angry’, ’sad’, ’happy’.

After this two-stage classification, the model labels input data as ’happy’, ’sad’, ’angry’, ’others’ example.

Furthermore, because of different class distributions in the datasets, I added an parameter T to the sent-others network to fit the model to development/test datasets classes distribution. This parameter specifies the minimum value that the softmax probability for ’sentiment’ class has to achieve to label input data as ’sentiment’. For predicting on the test dataset, I set the parameter T to 0.75, which was the value that achieved the best result on the validation dataset.

Parameter	Value		Tested Values
	sent-others	happy-sad-angry	
LAYERS_NUMBER	2	2	2
NEURONS_NUMBER	300, 300	250, 250	200, 200; 250, 250; 300, 300
BI-LSTM_INPUT_DROPOUT	0.3, 0.5	0.3, 0.5	0.2, 0.5; 0.3, 0.5
RECURRENT_DROPOUT	0.3, 0.5	0.3, 0.5	0.2, 0.5; 0.3, 0.5
FULL_CONNECTED_LAYER_DROPOUT	0.3	0.3	0.3, 0.4, 0.5
FULL_CONNECTED_LAYER_SIZE	120	120	100, 120
BATCH_SIZE	32	32	32
THRESHOLD_T	0.75	-	0.25, 0.55, 0.75

Table 1: The parameters of the proposed model.

Model	Validation(1000)			Test		
	sent-others	happy-sad-angry	2-stage	sent-others	happy-sad-angry	2-stage
ELMo	0.9456	**0.9700**	0.7511	0.9517	**0.9447**	0.7192
Word2Vec-310	0.9534	0.9641	0.7500	**0.9568**	0.9351	0.7180
Word2Vec-310 + ELMo	**0.9542**	0.9672	**0.7558**	0.9537	0.9422	**0.7278**

Table 2: Microaveraged F1 score of baselines models and the fusion model on the validation and the test datasets.

Table 1 presents tested model parameters and the best parameters set for the validation dataset.

3 Results

Table 2 presents microaveraged F1 score achieved on the test and the validation datasets for the proposed model containing also F1 scores for the first (sent-onthers network) and the second (happy-sad-angry network) classification stages. I compared the proposed fusion model against 2 baselines based on the static and contextualized models used in the proposed method, separately. Therefore the first baseline model uses 2-layer Bi-LSTM Neural Network which is learned on static, *Word2Vec* with affective features word representation, and the second one is a baseline model 2-layer BI-LSTM Neural Network as well but is learned on contextualized ELMo word representation embedding. The results of such baseline models allow better insight into the performance of the proposed fusion model.

The best results for 2-stage classification for the test dataset achieved the fusion model (0.7558 for validation and 0.7278 for the test dataset) despite the worse results in the sent-others and the happy-sad-angry classification stages. For the validation dataset, the proposed fusion model achieved the best score as well. The best result for 2-stage classification was better by about 1 percent from baselines results.

4 Conclusion

In this paper, we have presented the fusion model, a sentiment classifier that combines the features of static and contextualized word embedding. This approach achieved officially 0.7278 F1-score, ranking 47th out of 165 participants.

My results show that combining word embeddings can improve sentiment detection models based only on one, static or contextualized, embedding. The two-stage classification model can better insight to classification process and parameterized each stage separately.

References

Alexandra Balahur, Jesús M. Hermida, and Andrés Montoyo. 2011. Detecting implicit expressions of sentiment in text based on commonsense knowledge. In *Proceedings of the 2Nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, WASSA '11, pages 53–60, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christos Baziotis, Nikos Athanasiou, Alexandra Chronopoulou, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, Shrikanth Narayanan, and Alexandros Potamianos. 2018. NTUA-SLP at semeval-2018 task 1: Predicting affective content in tweets with deep attentive rnns and transfer learning. *CoRR*, abs/1804.06658.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of* *the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada. Association for Computational Linguistics.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

François-Régis Chaumartin. 2007. Upar7: A knowledge-based system for headline sentiment tagging. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 422–425, Prague, Czech Republic. Association for Computational Linguistics.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations. *CoRR*, abs/1707.06996.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *CoRR*, abs/1607.01759.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2873–2879. AAAI Press.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Rohit Saxena, Savita Bhat, and Niranjan Pedanekar. 2018. Emotionx-area66: Predicting emotions in dialogues using hierarchical attention network with sequence labeling. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, pages 50–55, Melbourne, Australia. Association for Computational Linguistics.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432, Lisbon, Portugal. Association for Computational Linguistics.

ParallelDots at SemEval-2019 Task 3: Domain Adaptation with feature embeddings for Contextual Emotion Analysis

Akansha Jain[1]
Paralledots, Inc.

Ishita Aggarwal[1]
Paralledots, Inc.

Ankit Narayan Singh[1]
Paralledots, Inc.

[1]akansha, ishita, ankit@paralleldots.com

Abstract

This paper describes our proposed system & experiments performed to detect contextual emotion in texts for SemEval 2019 Task 3. We exploit sentiment information, syntactic patterns & semantic relatedness to capture diverse aspects of the text. Word level embeddings such as Glove, FastText, Emoji along with sentence level embeddings like Skip-Thought, DeepMoji & Unsupervised Sentiment Neuron were used as input features to our architecture. We democratize the learning using ensembling of models with different parameters to produce the final output. This paper discusses comparative analysis of the significance of these embeddings and our approach for the task.

1 Introduction

Emotion Classification is more nuanced version of Sentiment Analysis. While Sentiment Analysis gives you a general idea about user experience by categorizing statements into positive or negative, Emotion Classification extracts specific attributes about each of these 2 categories. Contextual Emotion Classification needs to keep the context of an ongoing conversation to predict their emotional state and therefore comes with its own challenges. Detecting emotions has become a crucial part of understanding user generated content and to generate emotion aware responses. This paper describes our approach for SemEval 2019 Task 3: EmoContext. The task is Emotion Classification in the conversational scenario. Complete details about the task, evaluation and dataset can be found in paper released by organizers (Chatterjee et al., 2019). We use various state-of-the-art Machine Learning models and perform domain adaptation (Pan and Yang, 2010) from their source task to the SemEval EmoContext task. Our solution uses multiple types of feature embeddings viz Skip-Thought vectors (Kiros et al., 2015), Unsupervised Sentiment Neuron (Radford et al., 2017), DeepMoji's attention and last layer (Felbo et al., 2017) embedding along with Glove (Pennington et al., 2014), Emoji Embedding (Eisner et al., 2016) and FastText (Joulin et al., 2016). These feature embeddings are passed to a Deep Learning architecture. We train multiple models with different hyper-parameters. Finally, the results from each models are stacked together in an ensemble (Polikar, 2006). Our main approach for the literature survey was to look for similar research work used in previous SemEval tasks and other published state-of-the-art methodologies in the same domain. Infact, SemEval 2018 task of finding Affect in Tweet (Mohammad et al., 2018) demonstrates how detection of emotion plays an important role in understanding content as well as its creators. It was helpful to learn about how various architectures such as Siamese (Ghosh and Veale, 2018), CNNs (Khosla, 2018) and Deeply connected LSTMs (Wu et al., 2018) can be used to effectively learn emotional context of text. Methodologies such as ensembling (Polikar, 2006) and use of diverse features embeddings (Duppada et al., 2018) plays an important role when the data is limited, imbalanced and confusing to classify accurately. In this paper, we discuss our approach and experiments to solve this problem. The remainder of this paper is organized as follows. Section 2 explains the System Description and our analysis. Experiment setup and Results are discussed in section 3, followed by conclusion in the last section.

2 System Description

The following section describes our analysis of the task and data. We then discuss preprocessing steps, features used and system design with architecture flow.

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 185–189
Minneapolis, Minnesota, USA, June 6–7, 2019. ©2019 Association for Computational Linguistics

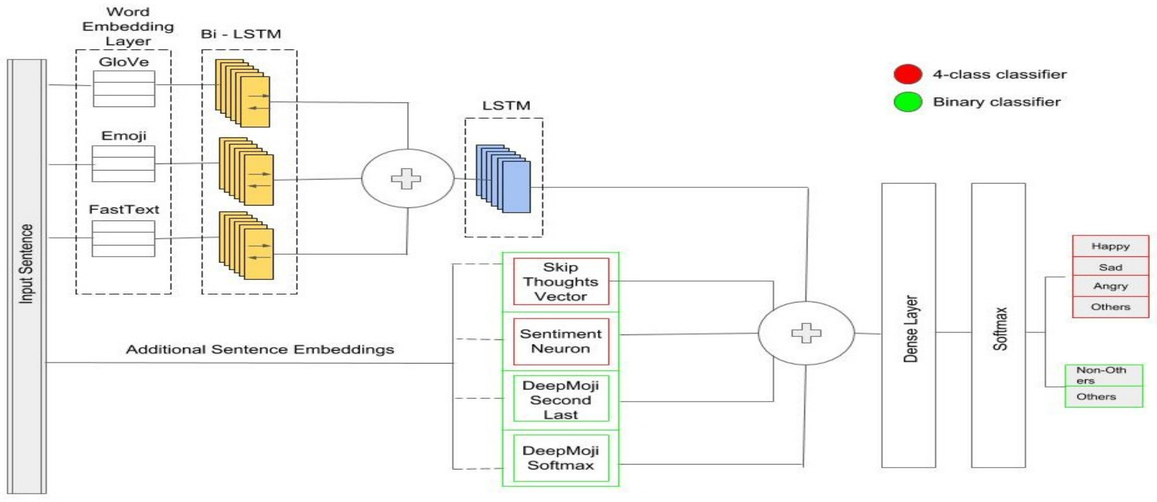


Figure 1: For each input sentence, word-level embeddings are passed to Bi-LSTMs, the output is concatenated and then passed through an LSTM. The sentence-level embeddings are then concatenated with LSTMs output and passed to the dense layer with softmax. First, the input is classified into others or non-others emotion category, the latter classified input is then passed to a second 4 class model for final classification.

2.1 Task and Data Analysis

The Data consists of 3 consecutive utterances in a conversation called turn1, turn2 and turn3. The task is to classify the emotion of the user on turn3. There are 4 labels which includes three emotions viz happy, sad, angry and others is used for emotionless label. After analysing the data we found some inconsistency like use of slangs (lol, xoxo), spelling errors(hellooo, frnd) and incomplete sentences. There is also difficulty in determining emotion because of ambiguity, for ex. I am not talking to u which can be either interpreted as sad or angry statement. Another important discovery shows that every labelled emotion is highly associated with the emojis used. The major issue with the data we faced was of class imbalance, 4% of data belonged to three emotion class and 88% belonged to the others category, which ultimately causes confusion between others and each emotion class. Hence, we decided to first classify the utterances into others or non others, and then further the non others into the 4 classes.

2.2 Pre Processing

We avoided removing stop words and lemmatization since it results in loss of information. We followed standard pre-processing steps: 1) All three utterances are concatenated using a placeholder <eos>. 2) All characters are converted to lowercase. 3) A contiguous sequence of emojis is split into individual emojis. 4) All repeated punctuation (???, ...) and white spaces are removed.

2.3 Features

During literature survey, we discovered different features that helped us capture an informal conversation as a whole. To tackle the ambiguity and other inconsistencies, we have used both word-level and sentence-level embeddings as input features to the model.

2.3.1 Word Level Embeddings

- Glove Embedding - We used 300 dimension Glove embedding to capture the general semantics of each word in the utterance.

- Emoji Embedding - Emoticons played a central role to understand the context of emotion in the text. We used a 300 dimension Emoji Embedding pre trained on large emoji corpus, to capture each emoji in the corpus which were being missed by Glove.

- FastText Embedding - We trained 300 dimension FastText embeddings on the training data to capture data specific semantics of the words.

2.3.2 Sentence Level Embeddings

- Skip-thought Vectors - We extracted 4800 dimension sentence embedding of the data using Skip-Thought vectors encoder, which capture generic sentence representation.

- Unsupervised Sentiment Neuron (USN)- We trained USN to obtain a 4096 dimension sentence embedding to capture the representation of sentiment in the text.

- DeepMoji - DeepMoji is trained on a huge corpus of 1.3 billion tweets for sentiment, emotion and sarcasm. Felbo et al.(Felbo et al., 2017) released the pre-trained model for the sole purpose of transfer learning for similar tasks. We extracted 2 feature sets on our dataset: DeepMoji Attention layer Embedding - 2304 dimension, DeepMoji Softmax Layer Embedding - 64 dimension.

2.4 Architecture

Our system comprises of 2 models. The first model is a binary classifier. It classifies the data into others and non-others. The second model is a 4-class classifier which further classifies non-others classes into all the 4 labels. For both models, vocab size is 20000, maximum sequence length is 100, word-level embedding size is 300, categorical cross entropy loss, and Adam optimiser is used, refer Figure 1. The type of additional sentence features to baseline are the only differentiating features to each model. The 4 class classifier exclusively takes Skip thought and USN sentence embeddings. On contrary, Binary classifier classifier takes Skip thought, USN, DeepMojis attention and softmax layer.

Baseline : For both models the basic architecture is same; All three word level embeddings, each learned by a Bi-LSTM, concatenated together is passed through LSTM to finally classify by a softmax dense layer.

2.5 Ensembling

We train five different classifiers for each of the model to perform stacked ensembling (Polikar, 2012). We use different configurations by changing value of learning rate, epoch size, LSTM dimensions and dropout rate to diversify the learning. The results from the models are given to meta classifier as input. The output of this meta model is treated as the final output of the system. Among different meta classifiers, our system achieved best results with logistic regression.

3 Experiments and Results

In this section, several experiments that were conducted to prove the effectiveness of our method are explained. All experiments and models concluded to benefit from (i) Pre-processing (ii) Emoji Embeddings (iii) Sentence Embeddings as extra features. The evaluation metrics used to compare results in the below section is micro F1 score. The metric used for evaluation on leaderboard score is micro averaged F1 of all three emotion classes.

3.1 Benchmarking on State-of-the-Art Architectures

We fine-tune 2 models on the EmoContext data viz. DeepMoji (Felbo et al., 2017) and Universal Language Model Fine-tuning (ULMFiT) (Howard and Ruder, 2018). Fine tuning DeepMoji on our data achieved an averaged F1 score of 0.65. The ULMFiT pretrains a language model (LM) on a large general-domain corpus and fine-tunes it on the target task. We deployed ULMFiT pretrained on standard Wikitext-103 (Merity et al., 2018) which limits the classifier for chat conversations data. It only achieved F-1 score of 0.56. Refer Table 1 for results.

3.2 Impact of Embeddings

Glove does not have embeddings for emoticons and removing emoji from the text results in emotional context loss. To overcome this challenge we employed a separate Emoji Embedding which played a crucial role to interpret the underlying emotion in a conversation as seen in Table 2. Emoji Embeddings turned out to give better results than replacing emojis with their description. To capture sentence representation for different aspects, sentence embeddings played an important role as explained in feature section. Results show major improvement in score with combination of sentence-embeddings and word embeddings, and in turn yields better performance than word-embeddings alone.

3.3 Impact of Extra Features

We also tried traditional approach to extract features from the data. They have shown to benefit the Machine Learning models in the past. In our case, including several sentence level features (number of words, number of special characters, number of emojis, average word length, readability index, compound valence score, 'negative valence score', neutral valence score, POS valence score, number of nouns, number of verbs) reduced the F-1 score below our baseline for test set. This is shown in Table 1. It is assumed the reason for the same is inconsistent and noisy data.

Model	**F-1 (avg)**	**Happy**	**Sad**	**Angry**	**Others**
ULMFiT	0.5600	0.47	0.63	0.59	0.93
DeepMoji	0.6551	0.62	0.71	0.65	0.93
4-Class classifier (M)	0.6954	0.68	0.71	0.70	0.95
M + Resampled data	0.6869	0.61	0.72	0.72	0.95
M + Extra features	0.7055	0.67	0.72	0.72	0.95
M + Ensemble (ME)	0.7128	**0.69**	0.72	0.73	0.95
Binary classifier (B) + ME	0.7180	0.68	0.72	0.74	0.95
(B + Ensemble) + ME*	**0.7201**	0.68	**0.75**	**0.74**	**0.96**

Table 1: Results with Additional Resources. * Final results for competition.

Embedding Feature Set	**F-1(avg)**	**Happy**	**Sad**	**Angry**	**Others**
Glove	0.5971	0.58	0.58	0.62	0.92
Glove + FastText	0.6657	0.66	0.70	0.64	0.93
Glove + FastText + Emoji Embeddings (WE)	0.6833	**0.70**	**0.71**	0.66	0.94
WE + Sentence Embeddings	**0.6954**	0.68	**0.71**	**0.70**	**0.95**

Table 2: Comparative Results on Embeddings.

Datasets	**F-1 (avg)**	**Happy**	**Sad**	**Angry**	**Others**
Emotion Push Chat Logs	**0.89448**	**0.87**	**0.86**	**0.96**	0.88
SemEval 2019 - Task 3	0.683386	0.70	0.71	0.66	**0.94**

Table 3: Experiments with Baseline.

3.4 Imbalance Data

To solve the problem of data imbalance, generating synthetic data is one of the many techniques. However, adding synthetic data which is made from down-sampling majority class and simultaneously up-sampling the minority classes didn't bring much improvement, and in this case even reduced the accuracy of the test result.

3.5 Final Classification

All the above experiments, after detailed analysis shows that major confusion exists between each emotion and 'others' label. This led us to the conclusion, to improve F1, a binary classification could be done first. Results also helped us to decide the number of classes for the second model to be 4 instead of 3 because 4 class model classifies the incorrect non-others back to the others category.

3.6 Effectiveness of Baseline

EmotionPush chat logs are conversations between friends on Facebook Messenger collected by an app called EmotionPush[1] (Chen et al., 2018). We take sample of this data in the same format as EmoContext. We trained our baseline architecture on both datasets. Comparative results in Table 3 shows how our simple baseline performs extremely well for EmotionPush texts, Moreover contrasting the subjective effect of noisy data on model performance for SemEval data.

4 Conclusion

Contextual Emotion detection, like any multi-class text classification requires powerful ability to comprehend the sentence in variety of aspects. In this contest, our model performed decent, scoring 72.01 on final leader board. For our method, emoji played very important role in understanding emotion in the text, and just by using Emoji Embedding we gained a significant improvement in F1. We proved how feature engineering can be very powerful on skewed and imbalanced data to capture contexts in NLP. We present a simple baseline of our model that gives commendable results for a general Emotion Classification scenario as proven for EmotionPush sample data.

[1]Participants consented to make their private conversations available for research purposes.

References

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Sheng-Yeh Chen, Chao-Chun Hsu, Chuan-Chun Kuo, Lun-Wei Ku, et al. 2018. Emotionlines: An emotion corpus of multi-party conversations. *arXiv preprint arXiv:1802.08379*.

Venkatesh Duppada, Royal Jain, and Sushant Hiray. 2018. Seernet at semeval-2018 task 1: Domain adaptation for affect in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 18–23. Association for Computational Linguistics.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. *arXiv preprint arXiv:1609.08359*.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524*.

Aniruddha Ghosh and Tony Veale. 2018. Ironymagnet at semeval-2018 task 3: A siamese network for irony detection in social media. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 570–575. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Sopan Khosla. 2018. Emotionx-ar: Cnn-dcnn autoencoder based emotion classifier. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, pages 37–44.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. An analysis of neural language modeling at multiple scales. *arXiv preprint arXiv:1803.08240*.

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 task 1: Affect in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 1–17. Association for Computational Linguistics.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Robi Polikar. 2006. Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, 6(3):21–45.

Robi Polikar. 2012. Ensemble learning. In *Ensemble machine learning*, pages 1–34. Springer.

Alec Radford, Rafal Józefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *CoRR*, abs/1704.01444.

Chuhan Wu, Fangzhao Wu, Sixing Wu, Junxin Liu, Zhigang Yuan, and Yongfeng Huang. 2018. Thu_ngn at semeval-2018 task 3: Tweet irony detection with densely connected lstm and multi-task learning. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 51–56. Association for Computational Linguistics.

E-LSTM at SemEval-2019 Task 3: Semantic and Sentimental Features Retention for Emotion Detection in Text

Harsh Patel
Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar
201701021@daiict.ac.in

Abstract

This paper discusses the solution to the problem statement of the SemEval19: EmoContext competition(Chatterjee et al., 2019b) which is "Contextual Emotion Detection in Texts". The paper includes the explanation of an architecture that I created by exploiting the embedding layers of Word2Vec and GloVe using LSTMs as memory unit cells which detects approximate emotion of chats between two people in the English language provided in the textual form. The set of emotions on which the model was trained was Happy, Sad, Angry and Others. The paper also includes an analysis of different conventional machine learning algorithms in comparison to E-LSTM.

1 Introduction

Emotions are the basic human quality that almost every human possesses. According, to a recent study by Glasgow University[1], human emotions can be divided into six basic classes which are happiness, sadness, anger, fear, surprise and disgust, surprise being the most difficult one as both positive and negative statements can lead to a sense of surprise. For example, the statement *Your application for CSE branch in Stanford University is accepted* is positive and it leads to surprise whereas the statement *Your brother met with an accident* is a negative statement which all leads to a surprise.

Problem Statement: *Given a text for three turn conversation, classify the emotion of the text in the following four categories - Happy, Sad, Angry, Others.*

[1]https://www.gla.ac.uk/news/archiveofnews/2014/february/headline_306019_en.html

[2]WhatsApp is used as a messaging platform to illustrate the three turn conversation approach

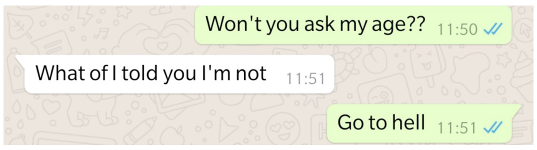

Won't you ask my age?? 11:50
What of I told you I'm not 11:51
Go to hell 11:51

Figure 1. Example of three turn conversation[2]

Detecting human emotions only from the text is very difficult as the emotions are a combination of the situation and the facial expressions of a person(Cowie et al., 2001). So, merely classifying it from the conversation is not a very accurate way.

In this paper, I have proposed an extended approach to the original model(Chatterjee et al., 2019a) which combines deep learning along with some techniques used in Natural Language Processing(NLP) using semantic and embedding approach (Franco-Salvador et al., 2018; Shivhare and Khethawat, 2012) called as "Emotion LSTM" or E-LSTM to detect emotions in the provided training set. The E-LSTM is a combination of both count-based and predictive techniques which are widely used in Natural Language Processing.

2 Approach

My approach in solving the given problem statement was to maintain the semantic and sentimental relationship among the words(Gupta et al., 2017). So, as shown in Figure 2, I modeled the architecture such that the lower part contains the embeddings for sentiment analysis whereas the upper part contains the embeddings for maintaining a semantic relationship. The embeddings are then passed onto a network of LSTM layers which memorize the relationship among the words. The output of the final LSTM cell is then flattened and is combined with the output of the LSTM cell in the other half. The combined matrix is then passed as an input to a dense network with two sub-levels

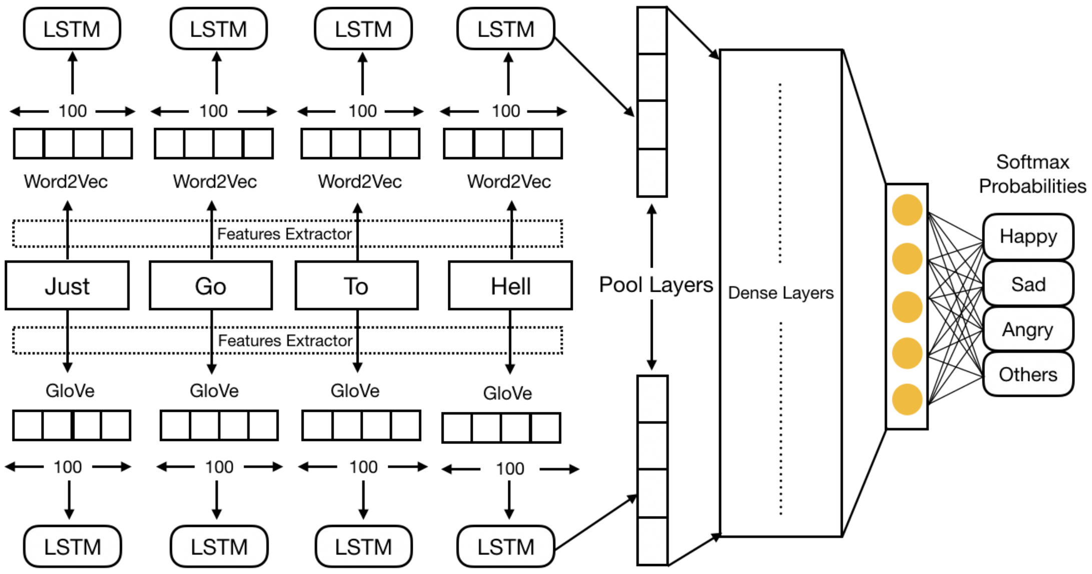


Figure 2. Architecture of E-LSTM model

Data	Labels	Happy	Sad	Angry	Others	Total
First Phase	#	4243	5463	5506	14948	30160
	%	14.07	18.11	18.26	49.56	100
Final Phase	#	142	125	150	2338	2755
	%	5.15	4.53	5.44	84.86	100

Table 1. Statistics of Training Dataset

whose output is then treated as a probability for the given four possible emotions using Softmax function.

2.1 Training Dataset

For the EmoContext SemEval-2019 Task 3, I was provided initially with a training dataset of about 30,000 entries containing 3 turn conversation and labels corresponding to each conversation. After successfully completing the first round, I was then provided with a final training dataset of about 2,700 entries. Statistics of both the datasets are shown in Table 1.

For the first phase, I proceeded with the provided dataset as a whole fro training whereas, in the second phase, I merged the provided new dataset with the dataset of Phase I and then used it for the model training.

2.2 Handling Repetition and Emoticons

After thoroughly analyzing the provided dataset, it was observed that emoticons were frequently used in the statements to describe the feeling or to end the statement. Similarly, special characters like . and * were also frequently used along with repetition. For example, You've got me blushing...☺ and Go to hell☹ statements. So, the first step of my data preprocessing was to remove the multiple instances of special characters and emoticons. So, the statement You've got me blushing...☺ after preprocessing became You've got me blushing. Other than normal preprocessing, the emoticons were also stored according to the sentence index in a dictionary and were used at the last step to verify if the predicted emotion matches partly or fully with the emotion depicted by used emojis using a weighted approach.

Word1,Word2	Word2Vec	GloVe
sad,:(	0.25	0.78
better,great	0.81	0.19

Table 2. Comparison of Word2Vec and GloVe embeddings in classifying relation among two words

2.3 Embedding Layers

The main challenge in the architecture of the model was to identify a proper embedding layer

#	Turn 1	Turn 2	Turn 3	True Label	Comments
1	You broke my heart	It was never mine to break !	See you are arrogant	sad	LSTM-Word2Vec failed because of word ”arrogant”
2	I like to cry	why are you crying	It was a joke	happy	Almost all model failed except E-LSTM model
3	You’re not giving me coupon nor photo	your phone is on mute hahahha	☹☹☹☹	sad	All the models failed but the last emoji comparing technique passed for the E-LSTM model
4	its only being childish	Your username is sad. ’-’ hug =/	how?	others	Counting based models failed becasue of negative words

Table 3. Qualitative Analysis of baseline models along with proposed E-LSTM model

to increase the model accuracy. The initial evaluations were passed only by using the baseline structure of GloVe embedding along with LSTM layers which proved to be costly as the micro F1 score that I got was comparatively less (about 0.57 for phase I and 0.61 for phase II) whereas the training time for significantly high. So, the accuracy of the model was improved through maintaining the semantic and syntactic features of statements intact by using the two novel types of research in the Natural Language Processing field which are Word2Vec(Word to Vector)(Mikolov et al., 2013; Rahmawati and Khodra, 2016) and GloVe(Global Vectors)(Baroni et al., 2014; Pennington et al., 2014) embedding layers. The Word2Vec embedding layers maintained the sentiments of the provided text whereas the GloVe embedding maintained the semantic feature of the text. As shown in Table 2, Word2Vec was better in classifying a relationship between sad and :(as it is a predictive model and was thus trained accordingly, whereas GloVe embedding was better in classifying relation between words better and great as the approach is completely based on counting i.e. counting involved in matrices operation.

2.4 Model Training

For training my E-LSTM model, I have used Keras library. As the data was limited, I have used the K-fold cross-validation method to train the model better. For training, I used K=5 i.e. 5 fold cross-validation. This number was chosen specifically after training on the data multiple times and comparing the obtained accuracy with the training time. The most optimal hyperparameters for my model were using CrossEntropy with Softmax as my loss function along with SGD(Stochastic Gradient Descent) as an optimizer with a learning rate of 0.003. For fully connected dense layers, I used a dropout of 0.3 to prevent over-fitting of the model. The batch size that I used while training the model was 800. Apart from hyperparameters, the main thing to note while concatenation of results obtained from the LSTM layers is the Leaky-ReLU layer that I have used. Reason being some negative input values which were completely discarded by normal ReLU layer.

3 Experimental Setup

In this section, I have described the statistics of my testing data along with a comparison of the obtained results with other models. I have also discussed some of the glitches that are evident in my model in the latter half.

3.1 Test Dataset

Similar to training dataset, test dataset was also provided in both the phases i.e. initial phase and final phase. But before the System-Design submission, one gold test dataset was also provided to test the model if it’s changed before paper submission. All the three test dataset files contained an index number and three turn conversation as their entry.

Model	Happy			Sad			Angry		
	Precision	Micro F1	F1	Precision	Micro F1	F1	Precision	Micro F1	F1
NB	45.4	56.32	50.27	74.22	70.1	72.10	43.21	38.21	40.57
SVM	75.21	32.1	45	94.45	66.66	78.16	92.11	62.21	74.26
CNN	64.3	49.32	55.82	76.21	70.12	73.04	74.12	49.44	59.32
CNN-GloVe	57.9	58.43	58.16	92.11	77.43	84.13	73.11	74.47	73.78
GloVe-LSTM	69.31	49.87	58	82.6	87.42	84.94	79.12	64.21	70.89
W2V-LSTM	75.42	45.55	56.8	84.32	78.12	81.1	80.2	64.34	71.4
E-LSTM	76.68	61.3	**64.47**	92.11	82.12	**86.83**	94.32	69.89	**80.29**

Table 4. Comparison of accuracy of different models ran on validation dataset of Task-2

3.2 Baseline Approaches

For comparison and proving my model better, I compared it with two different categories - 1. Machine Learning based and 2. Deep Learning based

For Machine Learning based baseline models, I have used Naive Bayes(NB) and Support Vector Machine(SVM). As the used models are inefficient with large datasets, so I used a subset of provided dataset to train them.

For Deep Learning based baseline models, I have used normal Convolutional Neural Networks(CNNs), CNN combined with Long Short Term Memory(LSTM) memory unit cells for data remembering, CNN combined with embedding layer of Global Vectors(GloVE) which maintains the sentiments in the text, LSTMs combined with GloVe embedding layer to find the sentiments among words and Word2Vect embedding layer combined with LSTMs which is used to maintain the semantic features in a statement. For all the deep learning baseline architectures, text in batches was given as input.

4 Results

As seen in the table Table 4, E-LSTM model outperformed all other models in both F1 score and average F1 score for all classes of emotion. Hence, it can be concluded that combining semantic and sentiment features of a statement can lead to better accuracy of emotion detection. It is also evident that Deep Learning models like CNNs, LSTMs, and RNNs are better than normal Machine Learning models like SVMs.

4.1 Qualitative Analysis

It is evident from Table 3 that E-LSTM model performed best as it tackled all the cases where counting based models when actual emotion is different from the words used in the conversation. Sentimental features also provided wrong results sometimes due to the predicted and true emotions being very close. The third entry in the table involves conversation which is highly contradicting from the true emotion. Thus, almost all the models failed in this type of case. But the verification of predicted emotion with the emoticons as described earlier saved the E-LSTM model from failing. Thus, the handcrafted features at the end of the model are very useful in this type of scenarios.

5 Conclusion

Evaluation of the given test data set shows that my model outperforms classical machine learning algorithms and also simple CNN and LSTM layers based models. Thus, it can be concluded that maintaining the semantic and syntactic relationship among words can be useful to identify emotions from texts accurately.

References

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 238–247.

Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. 2019a. Understanding emotions in text using deep learning and big data. *Computers in Human Behavior*, 93:309–317.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019b. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Roddy Cowie, Ellen Douglas-Cowie, Nicolas Tsapatsoulis, George Votsis, Stefanos Kollias, Winfried Fellenz, and John G Taylor. 2001. Emotion recognition in human-computer interaction. *IEEE Signal processing magazine*, 18(1):32–80.

Marc Franco-Salvador, Sudipta Kar, Thamar Solorio, and Paolo Rosso. 2018. Uh-prhlt at semeval-2016 task 3: Combining lexical and semantic-based features for community question answering. *arXiv preprint arXiv:1807.11584*.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations. *arXiv preprint arXiv:1707.06996*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Dyah Rahmawati and Masayu Leylia Khodra. 2016. Word2vec semantic representation in multilabel classification for indonesian news article. In *2016 International Conference On Advanced Informatics: Concepts, Theory And Application (ICAICTA)*, pages 1–6. IEEE.

Shiv Naresh Shivhare and Saritha Khethawat. 2012. Emotion detection from text. *CoRR*, abs/1205.4944.

ELiRF-UPV at SemEval-2019 Task 3: Snapshot Ensemble of Hierarchical Convolutional Neural Networks for Contextual Emotion Detection

José-Ángel González, Lluís-F. Hurtado, Ferran Pla
Departament de Sistemas Informàtics i Computació
Universitat Politècnica de València.
Camí de Vera, sn
46022, València
{jogonba2, lhurtado, fpla}@dsic.upv.es

Abstract

This paper describes the approach developed by the ELiRF-UPV team at SemEval 2019 Task 3: Contextual Emotion Detection in Text. We have developed a Snapshot Ensemble of 1D Hierarchical Convolutional Neural Networks to extract features from 3-turn conversations in order to perform contextual emotion detection in text. This Snapshot Ensemble is obtained by averaging the models selected by a Genetic Algorithm that optimizes the evaluation measure. The proposed ensemble obtains better results than a single model and it obtains competitive and promising results on Contextual Emotion Detection in Text.

1 Introduction

Emotion Detection problem arises in the context of conversational interactions, among two or more agents, when one agent is interested in knowing the emotional state of other agent involved in the conversation. The detection of emotions is a difficult task when the content is expressed by using only text, due to the lack of facial and hand gesture expressions, voice modulations, etc. Moreover, the task becomes more complex if the detection of emotions is applied only on a short piece of text without including context. This is because the context can act as an emotion modifier of a given turn in the conversation.

Although, researchers mainly focus on emotion detection on text in absence of context (Mohammad et al., 2018) (Klinger et al., 2018), tipically extracted from social media, recently, there are few works that approach the emotion detection in conversations by using context information (Hazarika et al., 2018b) (Majumder et al., 2018) (Hazarika et al., 2018a). These contextual systems work on long conversations where different users are involved and they use multimodal data, specifically, text, audio and video in order to address the emotion detection problem on large multi-party conversations.

In this work, we present an approach to the Semeval 2019 Task 3: Contextual Emotion Detection in Text (Chatterjee et al., 2019). This task is a simplification of the text emotion detection problem on conversations where each conversation have only three utterances. Only two different users are involved in each conversation, where the first and third turn corresponds to the first user and the second turn corresponds to the second user. The goal of this tasks is to predict the emotion of the third turn. We propose a Snapshot Ensemble (SE) of 1D Hierarchical Convolutional Neural Networks (HCNN) trained to extract useful information from 3-turn conversations. Our system was designed following some ideas of (Morris and Keltner, 2000) and (Majumder et al., 2018). Concretely, we consider the inter-turn and self-turn dependencies (Morris and Keltner, 2000) along with the context given by the preceding utterances (Majumder et al., 2018) to determine the emotion of a given turn.

2 System Description

2.1 Preprocessing

For the tokenization process, our system used TweetTokenizer from NLTK (Loper and Bird, 2002). In addition, we performed some other actions. All the text was transformed to lowercase. Multiple spaces were converted to a single space. Urls were replaced by the tag "url". We transformed multiple instances of punctuation marks in a single one (e.g., "???" $\rightarrow$ "?"). In order to extract semantic representations of the unicode emojis, they are replaced by their description using the Common Locale Data Repository (CLDR) Short Name (e.g., 🤩 $\rightarrow$ "grinning face with star eyes"). Moreover, non relevant and common words are removed from these descriptions ("grinning face with star eyes" $\rightarrow$ "grinning star eyes").

2.2 Word Embeddings

It is well known that word embeddings (WE) learned from the same domain of a downstream task usually lead to obtain better results than those obtained using general domain WE. Due to the fact that we did not have sentences of the task to learn word embeddings from them, we used embeddings learned from Twitter posts because we considered that the characteristics of tweets are similar to the task language. Both of them have a noisy nature and they share common features of the internet language (slang, letter homophones, onomatopoeic spelling, emojis, lexical errors, etc.). Therefore, we used 400-dimensional WE obtained from a skip-gram model trained with 400 million tweets gathered from 1/3/2013 to 28/2/2014 (Godin et al., 2015).

2.3 Hierarchical Convolutional Neural Networks

We considered several characteristics of the task in order to design our system. First, the utterances are short and there are many short-term dependencies among these words. Therefore, we propose to use 1D Convolutional Neural Networks (Kim, 2014) (CNN) to extract a rich semantic representation of each utterance. Second, the conversations are composed only by 3 utterances, for that reason, it is not required to uses models with high capacity to learn long contexts. Thus, we propose to use another CNN on top of the first CNN that extracts sentence representations, in order to obtain representation of conversations. We called this approach Hierarchical Convolutional Neural Networks (HCNN) following the work of (Yang et al., 2016).

As input to the model, each utterance j (composed by a maximum of N words) in a conversation i is arranged in a matrix $M_j \in \mathbb{R}^{N\times d}$, where each row corresponds with a word in the utterance j, represented by using d-dimensional WE. As each conversation is a sequence of three utterances, these conversations are arranged in a 3-dimensional matrix where each channel j is the representation of the utterance j in the conversation, i.e. for the conversation i, $M_i \in \mathbb{R}^{3\times N\times d}$. On all the matrices of M_i, 1D Dropout (Srivastava et al., 2014) was used to augment the dataset, by deleting words of each utterance with $p = 0.3$.

Given the representation of the conversation i, M_i, for each utterance independently, a CNN with kernels of different sizes is applied in order to obtain a composition of word embeddings that can extract semantic/emotional properties from each utterance. At this first level, we use $f_1 = 256$ kernels of sizes $\{2, 4, 6\}$ and their weights are shared among the three channels. From that, for each utterance, three new matrices are obtained. These matrices capture relevant features for each kernel size and utterance. These features are pooled into a vector by using 1D Global Max Pooling (GMP).

The resulting three vectors from the previous level were concatenated as rows to obtain a matrix representation of the conversation i composed by the CNN map of its sentences, $Wi \in \mathbb{R}^{3\times f_1}$. We considered that conversation features could be relevant for the task. At this level, in order to extract these relevant features and following the ideas in (Morris and Keltner, 2000) (Majumder et al., 2018), the system is intended to take into account the context and potentially the emotions given by preceding utterances to determine the emotion expressed by the last utterance. To do this, a CNN with $f_2 = 256$ kernels of sizes $\{1, 2, 3\}$ were used. The size of the filters is crucial to understand what features the system is capable to learn.

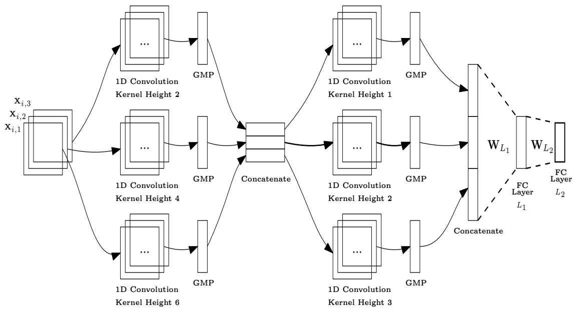
Concretely, 3-size kernels: semantic/emotional features over all the contexts (full conversation); 2-size kernels: inter-turn features and semantic/emotional features of preceding and later utterances given a context of two utterances; 1-size kernels: self-turn features and semantic/emotional features of each utterance independently.

On the output maps of this second CNN[1], GMP is used in order to extract the most relevant features from each dimension and the resulting vectors are concatenated. Later, a fully connected layer L_1 with 512 neurons is used to fuse the concatenated vectors. Finally, to obtain a probability distribution over $\mathbb{C}$ classes ({happy, sad, angry, others}) we use a softmax fully connected layer L_2. Figure 1 shows the proposed model architecture.

2.4 Snapshot Ensemble

Generally, ensemble models outperform single models in similar tasks (Duppada et al., 2018) (Rozental et al., 2018). Therefore, we decided

[1]After all the CNN layers (at two levels), BatchNormalization, LeakyReLU and Dropout are applied



$x_{i,3}$
$x_{i,2}$
$x_{i,1}$
1D Convolution Kernel Height 2
1D Convolution Kernel Height 4
1D Convolution Kernel Height 6
GMP
Concatenate
1D Convolution Kernel Height 1
1D Convolution Kernel Height 2
1D Convolution Kernel Height 3
Concatenate
W_{L_1}
W_{L_2}
FC Layer L_1
FC Layer L_2

Figure 1: Hierarchical Convolutional Neural Networks.

to use ensemble methods instead of trying different architectures. We used the ideas of Snapshot Ensemble (SE) (Huang et al., 2017) to combine HCNN trained until reaching good and diverse local minima by using SGD and a cosine learning rate with $T = 24$ training iterations, $M = 6$ learning cycles, and initial learning rate $alpha = 0.4$.

From this training method, we took 24 snapshots (one for each training iteration). From the set of snapshots $S = \{s_i \ / \ 1 \leq i \leq 24 \wedge s_i : \mathbb{R}^{3 \times N \times d} \rightarrow \mathbb{R}^{\mathbb{C}}\}$, we generate 4 different systems:

1. Best snapshot of all iterations

$$f_1 = \underset{s_i}{\text{argmax}} \ \mu F_1(s_i(x), y) \qquad (1)$$

2. Average of all snapshots

$$f_2 = \frac{1}{|S|} \sum_{s_i \in S} s_i(x) \qquad (2)$$

3. Average of best snapshot at each learning cycle

$$f_3 = \frac{M}{T} \sum_{i=0}^{\frac{T}{M}-1} \underset{s_i \in S[\frac{T}{M} \cdot i, \frac{T}{M} \cdot (i+1)]}{\text{argmax}} \mu F_1(s_i(x), y) \qquad (3)$$

4. Average of genetic selected snapshots

$$f_4 = \frac{1}{|g(S)|} \sum_{s_i \in g(S)} g(S)_i \ s_i(x) \qquad (4)$$

where x and y are the input and the target, respectively, and $g(S)_i$ is the decision of a genetic algorithm to include the snapshot s_i in the ensemble. We used this method in order to discretely select ($g(S)_i \in \{0, 1\}$) what snapshots are well-suited for the final averaging ensemble which tries to optimize μF_1. The genetic algorithm (Mitchell, 1998) starts with a population of 400 individuals, they are crossed by using two point crossover, mutated with flip bit and selected by using tournament selection during 100 generations. Moreover, this algorithm addresses a multi-objective problem, it must to reach combinations of snapshots whose averaged predictions yield to high values of μF_1 while minimizing the number of models in the ensemble (the final genetic ensemble is composed by 6 system, i.e. as many systems as learning cycles) These decisions were taken in order to reduce the overfiting risk during the learning of the ensemble i.e. we prioritize simpler ensembles which are composed by discretely selected snapshots.

3 Analysis of Results

In order to evaluate different configurations of our system we used the development set given by the task organizers. On this development set, ablation analysis on single HCNN was carried out in order to observe if the input Dropout and the incorporation of L_1 layer yield to better results (the capacity of HCNN must be greater when including both techniques). The results of this ablation analysis are shown in Table 1.

System	$\boldsymbol{\mu P}$	$\boldsymbol{\mu R}$	$\boldsymbol{\mu F_1}$
Vanilla	70.65	75.06	72.79
Dropout	71.78	**76.26**	73.95
$\boldsymbol{L_1}$	72.36	75.23	73.84
Dropout + $\boldsymbol{L_1}$	**75.42**	75.78	**75.60**

Table 1: Ablation analysis of input Dropout and L_1 layer on HCNN (development set)

Vanilla system is a single HCNN without input Dropout neither the L_1 layer. It can be observed that, the systems with Dropout and L_1 outperformed the **Vanilla** version of HCNN in terms of μP, μR and μF_1. In terms of μP, the systems which incorporate L_1 achieved better results. However, although **Dropout +** $\boldsymbol{L_1}$ obtained the best improvement in terms of $\boldsymbol{\mu P}$, the highest $\boldsymbol{\mu R}$ was obtained using only **Dropout**. This could indicate that data augmentation could be useful to increase the $\boldsymbol{\mu R}$ but it is required more network capacity to handle this augmentation in order to increase also the $\boldsymbol{\mu P}$.

These results were obtained by using a single HCNN with *adam* as update rule (Kingma and Ba, 2014) with default learning rate. However, the SE training mode with Vanilla SGD and cosine learning rate, along with the proposed ensemble generation, allows the **Dropout + L_1** system to reach better results (Table 2).

Ensemble	μP	μR	μF_1
Best snapshot (single)	74.82	78.41	76.58
Average All	74.38	77.93	76.18
Best per Cycle	75.29	77.45	76.35
Genetic Average	**75.73**	**80.09**	**77.85**

Table 2: Results on development set with several SE of HCNN.

In this case, the best single model (**Best snapshot**) obtained in the SE training mode, provided higher μR than **Dropout + L_1** at the expense of a reduction in μP. This improvement of 3 points of μR yields also an increase of the μF_1 measure.

Among the ensembles, only **Genetic Average** improves the **Best snapshot** and **Dropout + L_1** systems in all the metrics. This is due to a big increase in μR. This suggests that it is possible to improve the μF_1 results by balancing μP and μR.

The other ensembles obtain lower results in terms of μR and μF_1 than **Best snapshot**, which is a single model. Moreover, all SE (including **Best snapshot**) except **Genetic Average** are less accurate (lower μP) than **Dropout + L_1**. However, all of them improved considerably the μR.

Due to the SE HCNN models generally outperformed the best single model **Dropout + L_1** in terms of μF_1 on the development set, we submitted all these systems to be evaluated on the test set. The results are shown in Table 3. It can be seen that the best system is **Genetic Average**, the same behavior observed on the development set. Although **Best snapshot** is more accurate than the ensembles (higher μP), two of the three ensembles yields better results μF_1. Moreover, a big degradation in the results are observed, all systems goes from 77 μF_1 on the development set, to 74 μF_1 on the test set.

System	μP	μR	μF_1
Best snapshot (single)	**75.69**	72.60	74.11
Average All	73.15	75.00	74.07
Best per Cycle	73.27	75.12	74.18
Genetic Average	73.43	**75.72**	**74.56**

Table 3: Results on test set with several SE of HCNN.

Table 4 shows the results of our best system (**Genetic Average**) at class level. The worse classified classes in terms of F_1 were Angry and Happy.

Class	P	R	F_1
Angry	68.73	78.19	73.16
Happy	75.19	69.37	72.16
Sad	77.82	80.00	78.90

Table 4: Results at class level of **Genetic Average** on test set.

4 Conclusion and Future Work

In this paper, we have presented Snapshot Ensembles of Hierarchical Convolutional Neural Networks to address the Semeval 2019 Task 3: Contextual Emotion Detection in Text. Our system is based on the use of a Genetic Algorithm in order to ensemble different snapshots of the same model. This ensemble outperformed single models and also classical snapshot ensembles, obtaining competitive results in the addressed task.

Due to the fact that in the proposed system, the semantic and emotional information is only provided by the representation of the words and the utterances, as future work we plan to study different word and sentence embeddings. It would be also interesting to incorporate other emotional or sentiment features such as: Sentiment Unit (Radford et al., 2017), DeepMoji (Felbo et al., 2017), Sentiment Specific WE (Tang et al., 2014); or polarity lexicons. Moreover, we are also interested in work with more powerful word embeddings such as BERT (Devlin et al., 2018) in order to incorporate a richer semantic word representation.

Acknowledgments

This work has been partially supported by the Spanish MINECO and FEDER founds under project AMIC (TIN2017-85854-C4-2-R) and the GiSPRO project (PROMETEU/2018/176). Work of José-Ángel González is also financed by Universitat Politècnica de València under grant PAID-01-17.

References

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Venkatesh Duppada, Royal Jain, and Sushant Hiray. 2018. Seernet at semeval-2018 task 1: Domain adaptation for affect in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 18–23. Association for Computational Linguistics.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625. Association for Computational Linguistics.

Fréderic Godin, Baptist Vandersmissen, Wesley De Neve, and Rik Van de Walle. 2015. Multimedia lab $@$ acl wnut ner shared task: Named entity recognition for twitter microposts using distributed word representations. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 146–153. Association for Computational Linguistics.

Devamanyu Hazarika, Soujanya Poria, Rada Mihalcea, Erik Cambria, and Roger Zimmermann. 2018a. Icon: Interactive conversational memory network for multimodal emotion detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2594–2604. Association for Computational Linguistics.

Devamanyu Hazarika, Soujanya Poria, Amir Zadeh, Erik Cambria, Louis-Philippe Morency, and Roger Zimmermann. 2018b. Conversational memory network for emotion recognition in dyadic dialogue videos. In *NAACL-HLT*.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. 2017. Snapshot ensembles: Train 1, get m for free. *CoRR*, abs/1704.00109.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Roman Klinger, Orphee De Clercq, Saif Mohammad, and Alexandra Balahur. 2018. Iest: Wassa-2018 implicit emotions shared task. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 31–42. Association for Computational Linguistics.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA. Association for Computational Linguistics.

Navonil Majumder, Soujanya Poria, Devamanyu Hazarika, Rada Mihalcea, Alexander F. Gelbukh, and Erik Cambria. 2018. Dialoguernn: An attentive rnn for emotion detection in conversations. *CoRR*, abs/1811.00405.

Melanie Mitchell. 1998. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA.

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 task 1: Affect in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 1–17. Association for Computational Linguistics.

Michael W. Morris and Dacher Keltner. 2000. How emotions work: The social functions of emotional expression in negotiations. *Research in Organizational Behavior*, 22:1 – 50.

Alec Radford, Rafal Józefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *CoRR*, abs/1704.01444.

Alon Rozental, Daniel Fleischer, and Zohar Kelrich. 2018. Amobee at iest 2018: Transfer learning from language models. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 43–49. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1555–1565. Association for Computational Linguistics.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *HLT-NAACL*.

EmoDet at SemEval-2019 Task 3: Emotion Detection in Text using Deep Learning

Hani Al-Omari
alomarihani1997@gmail.com

Malak Abdullah
mabdullah@just.edu.jo

Nabeel Bassam
nabeelbassam98@gmail.com

Department of Computer Science
Jordan University of Science and Technology
Irbid, Jordan

Abstract

Task 3, EmoContext, in the International Workshop SemEval 2019 provides training and testing datasets for the participant teams to detect emotion classes (Happy, Sad, Angry, or Others). This paper proposes a participating system (EmoDet) to detect emotions using deep learning architecture. The main input to the system is a combination of Word2Vec word embeddings and a set of semantic features (e.g. from AffectiveTweets Weka-package). The proposed system (EmoDet) ensembles a fully connected neural network architecture and LSTM neural network to obtain performance results that show substantial improvements (F1-Score 0.67) over the baseline model provided by Task 3 organizers (F1-score 0.58).

1 Introduction

The past decades have seen an explosive growth of user-generated content through social media platforms. People are expressing online their feelings and opinions on a variety of topics on a daily basis. Tracking and analyzing public opinions from social media can help to predict certain political events or predicting people's attitude towards certain products. Therefore, detecting sentiments and emotions in text have gained a considerable amount of attention(Mohammad et al., 2018). Researchers and scientists in different fields considered this a promising topic (Abdullah et al., 2018; Liu, 2012). Many machine learning approaches have been used to detect and predict emotions and sentiments. Recently, the deep neural network (DNN) is attracting more researchers as they have been benefited from the high-performance graphics processing unit (GPU) power (Abdullah et al., 2018; Dos Santos and Gatti, 2014).

The shared task (Task 3: "EmoContext") in SemEval-2019 workshop has been designed for understanding emotions in textual conversations (Chatterjee et al., 2019). In this task, the participants are given a textual dialogue i.e. a user utterance along with three turns of context. The participant teams have to classify the emotion of user utterance as one of the emotion classes: Happy, Sad, Angry or Others. Further details about Task 3 and the datasets appear in Section 3.

This paper describes our team approach to detect and classify emotions. The input has been represented as word vectors (Mikolov et al., 2013b) and a set of different features which are applied to different neural network architecture to obtain the results. The performance of the system shows substantial improvements F1-Score over the baseline model provided by Task 3 organizers.

The remainder of this research paper is organized as follows: Section 2 gives a brief overview of existing work on social media emotion and sentiment analyses. Section 3 presents the requirements of SemEval Task3 and examines our proposed system to determine the presence of emotion in conversational text. Section 4 summarizes the key findings of the study and the evaluations and concludes with future directions for this research.

2 Related Work

Defining and theorizing emotions had been investigated by several psychology researchers (Plutchik, 1990; Ekman and Keltner, 1997). The basic emotions according to Ekman (Ekman and Keltner, 1997) had been identified as anger, disgust, fear, happiness, sadness, and surprise. A little corpus exists for emotion labeling with text. Recently, several shared tasks and challenges had been introduced for detecting the intensity of emotion felt by the speaker of a tweet (Mohammad et al., 2018; Strapparava and Mihalcea, 2007). A group of researchers (Mohammad and Bravo-Marquez, 2017) introduced the WASSA- 2017 shared task of detecting the intensity of emotion felt by the speaker of a tweet. The previous Semeval Task1 (Mohammad et al., 2018) also introduced a dataset (annotated tweets) for emotion detection. The state-of- the-art systems in the previous competitions used different approaches of ensembling different deep neural network-based models, representing tweets as word2vec/doc2vec embedding vectors and extracting semantic features. Our system is using word2vec embedding vectors (Mikolov et al., 2013a) and extracted features using a Weka package, AffectiveTweet, (Bravo-Marquez et al., 2014), also extracting embedding from the text using deeMoji model (Felbo et al., 2017).

3 Our Approach

Our system has the ability to determine the emotion (Happy, Sad, Angry and Other) in English textual dialogue with F1-Score over 0.67. Figure 1 shows the general structure of the system. More details for the systems components are shown in the following subsections: Section 3.1 describes the systems input and preprocessing step. Section 3.2 lists the extracted feature vectors, and Section 3.3 details the system's architecture of neural networks. Section 3.4 discusses the output details.

3.1 Input and Preprocessing

The shared task (Task 3: EmoContext) provides training and testing datasets to be used by all participants. The number of training and testing datasets for each emotion can be shown in Table 1.

	Train Data	**Test Data**
Anger	5656	298
Happy	4385	284
Sad	5588	250
Other	17286	4677
Total	32915	5509

Table 1: Training and Testing Datasets

The training dataset contains 5 columns:

ID - Contains a unique number to identify each training sample.

Turn 1 - Contains the first turn in the three turn conversation, written by User 1.

Turn 2 - Contains the second turn, which is a reply to the first turn in conversation and is written by User 2.

Turn 3 - Contains the third turn, which is a reply to the second turn in the conversation, which is written by User 1.

Label - Contains the human-judged label of Emotion of Turn 3 based on the conversation for the given training sample. It is always one of the four values - 'happy', 'sad, 'angry' and 'others'.

For testing dataset, the 5th column - 'Label' is absent. See Table 2 for more clarification.

The prepossessing methods applied for the data include converting the text into lower case, stemming the words and removing of extraneous white spaces. Punctuation has been treated as individual words (".,?!:;()[]#@'). It's worth mentioning that removing stop-words dissolved the meaning of the sentence, therefore we didn't remove them.

3.2 Feature Vector

We have explored different features to represent each turn and the concatenated turns. Our approach extracts feature vectors from texts with a total of 2753 dimensions (Check Table 3). We have applied the same methods for each turn and the concatenated turns.
Each turn is represented as a 300-dimensional vector using the pretrained word2vec embedding model that is trained on Google News (Mikolov et al., 2013a). We have used the summation technique to represent every turn or conversation. In addition to that, each turn/conversation is represented as 145 dimensional vectors by concatenating three vectors obtained from the AffectiveTweets Weka-package bravo2014meta, mohammad2017wassa, 43 features have been extracted using the TweetToLexiconFeatureVectorattribute that calculates attributes for a tweet using a variety of lexical resources; two-dimensional vector using the sentiments strength feature from the same package, and the final 100 dimensional vectors is obtained by vectorizing the tweets to embeddings attribute also from the same package. We have also extracted 2302 dimensions vector using the attention layer of DeepMoji model (Felbo et al., 2017). Finally, we have used the NRC Valence, Arousal, and Dominance Lexicon to extract the last 4-dimensional

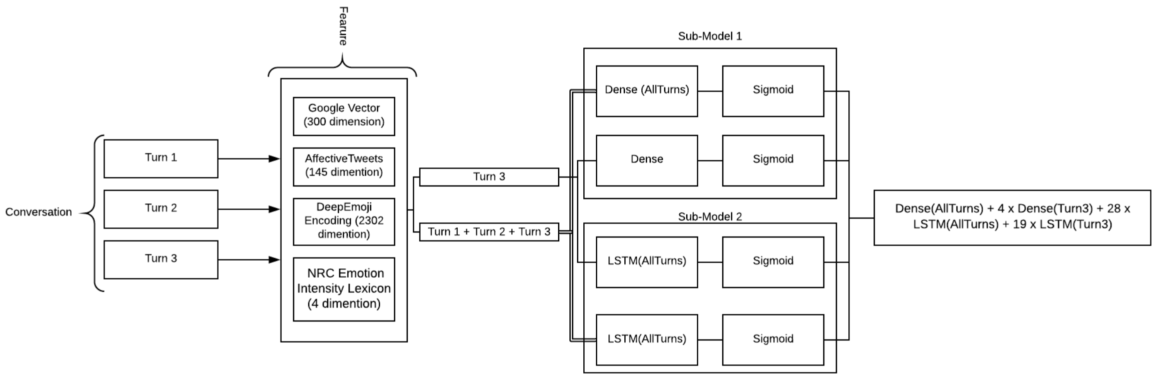

Feature
Conversation
Turn 1
Turn 2
Turn 3
Google Vector (300 dimension)
AffectiveTweets (145 dimention)
DeepEmoji Encoding (2302 dimention)
NRC Emotion Intensity Lexicon (4 dimention)
Turn 3
Turn 1 + Turn 2 + Turn 3
Sub-Model 1
Dense (AllTurns)
Sigmoid
Dense
Sigmoid
Sub-Model 2
LSTM(AllTurns)
Sigmoid
LSTM(AllTurns)
Sigmoid
Dense(AllTurns) + 4 x Dense(Turn3) + 28 x LSTM(AllTurns) + 19 x LSTM(Turn3)

Figure 1: The architecture of our approach

id	Turn1	Turn2	Turn3	label
156	You are funny	LOL I konw that.	:)	happy
187	Yeah exactly	Like you said, like brother like sister ;)	Not in the least	others

Table 2: Examples of datasets format

vector to represent Anger, fear, sadness, and joy (Mohammad, 2018).

	Dimension
Word2Vec	300
AffectiveTweets	145
DeepMoji	2302
NRC	4
Total	2753

Table 3: Feature vectors

3.3 Network Architecture

Knowing that Deep Neural Networks (DNN) is showing significant improvements over traditional Machine Learning (ML) based approaches on classification tasks(LeCun et al., 2015). This derives more researchers to apply it recently for detecting sentiments and emotions. The standard Recurrent Neural Network (RNN) is distinguished from Feed-forward network with a memory. A special kind of RNNs are Long Short-Term Memory Network (LSTM) (Hochreiter and Schmidhuber, 1997), which is composed of a memory cell, an input gate, an output gate and a forget gate.

The architecture of our system consists of two sub-models that use both: feed-forward (Dense) and LSTM. For our first sub-Model, the Input 2753-dimensional vector feeds a fully connected neural network with three dense hidden layers of 500, 200 and 80 neurons for each layer. The activation function for each layer is ReLU (Goodfellow et al., 2013). Two dropouts have been added to this sub-model, which are 0.3 and 0.2 after the first and the second layers. The output layer consists of 4 sigmoid neurons to predict the class of emotion in each conversation. For optimization, we use Stochastic Gradient Descent (SGD) optimizer (lr=.001, decay=1×10^{-6}, and momentum = 0.9) augmenting for MSE loss function and ACCURACY metrics. We have also saved the output predictions weights to predict the testing data sets. The fit function uses number of epochs = 60, batch size=32, validation split= 33%.

In the second sub-model, the same 2753-dimensional vector feeds an LSTM by using an embedding layer of 500-dimensions. The LSTM layer consists of 300 neurons with using Dropout 0.3 after the LSTM layer to avoid over-fitting. A dense layer with 200 neurons is added and followed by four sigmoid neurons to predict the emotion class in each conversation. For optimization, we use the same method as the first sub-model. We have also used early stopping technique to get the best result. finally, we have saved the output prediction weights to predict the testing data sets. The fit function uses number of epochs = 80, batch size=8, validation split= 33%.

Formula	**micro-F1**
$Dense(Turn1) + 2 \times Dense(Turn3)$	0.605355
$Dense(Turn1) + 2 \times Dense(Turn3) + 2 \times Dense(All)$	0.618162
$2 \times Dense(All) + 3 \times Dense(Turn3) + 3 \times LSTM(All)$	0.626
$Dense(All) + part3 + 3 \times LSTM(All)$	0.636
$Dense(All) + part3 + 10 \times LSTM(All)$	0.656165
$Dense(All) + 4 \times Dense(Turn3) + 28 \times LSTM(All) + 19 \times LSTM(Turn3)$	0.6714

Table 4: Weight Ensembling

Conversation	**Turn 1**	**Turn 2**	**Turn 3**
micro-F1	0.26379	0.08435	0.58920

Table 5: Using Sub-model 1 - Dense Layer

Conversation	**Turn 1**	**Turn 2**	**Turn 3**
micro-F1	0.20734	0.13543	0.44364

Table 6: Using Sub-model 1 - Dense Layer plus removing 70% of others randomly

System	**Epoch**	**micro-F1**
LSTM (All Conversation)	40	0.5376
LSTM (All Conversation)	80	0.6094
LSTM (All Conversation)	114	0.5096
Dense (All Conversation)	60	4677

Table 7: Best Epoch for both LSTM and Dense

3.4 Output and result

In the beginning, we have analyzed all the conversation (turn1 + turn2 + turn3) using both sub-model systems. We have noticed that the third turn of the conversation provides better predictions of emotion's class, see in Table 5. Removing 70% of the others randomly in the training data set led to bad predictions so we didn't apply this method, see Table 6. One of the key findings is noticing that LSTM gives better prediction than the feed-forward system for the whole conversation, see the result in Table 7. For the final stage, we have combined both sub-models results to produce a real value number between 0 and 1. It has shown that the second sub-model gives higher accuracy than the first sub-model. Applying different amount of weights for four prediction led us to find out that the correct formula for our system using turn3 alone as a sub-model and All-Conversation prediction and from the second sub-Model turn3 and All-Conversation and combining them together in a formula, It showed a higher F1-Score equal to 0.67. Also, it's worth mentioning that we used Grid Search to find the best parameters for the formula (Check Table 4).

4 Conclusion

In this paper, we have presented our system EmoDet that uses deep learning architectures for detecting the existence of emotions in a text. The performance of the system surpasses the performance of the baselines model indicating that our approach is promising. In this system, we uses word embedding models with feature vectors extracted using the AffectiveTweets package and Deepmoji model. These vectors feed different deep neural network architectures, feed-forward and LSTM, to obtain the predictions. We use the SemEval-2019 Task 3s datasets as input for our system and show that EmoDet has a high proficiency in detecting emotions in a conversational text and surpasses the F1-score Baseline models performance, which is provided by the SemEval-Task 3 organizers.

References

Malak Abdullah, Mirsad Hadzikadicy, and Samira Shaikhz. 2018. Sedat: Sentiment and emotion detection in arabic text using cnn-lstm deep learning. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 835–840. IEEE.

Felipe Bravo-Marquez, Marcelo Mendoza, and Barbara Poblete. 2014. Meta-level sentiment models for big social data analysis. *Knowledge-Based Systems*, 69:86–99.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Cicero Dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78.

Paul Ekman and Dacher Keltner. 1997. Universal facial expressions of emotion. *Segerstrale U, P. Molnar P, eds. Nonverbal communication: Where nature meets culture*, pages 27–46.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524*.

Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. *arXiv preprint arXiv:1302.4389*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*, 521(7553):436.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 task 1: Affect in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 1–17.

Saif M. Mohammad. 2018. Word affect intensities. In *Proceedings of the 11th Edition of the Language Resources and Evaluation Conference (LREC-2018)*, Miyazaki, Japan.

Saif M Mohammad and Felipe Bravo-Marquez. 2017. Wassa-2017 shared task on emotion intensity. *arXiv preprint arXiv:1708.03700*.

Robert Plutchik. 1990. Emotions and psychotherapy: A psychoevolutionary perspective. In *Emotion, psychopathology, and psychotherapy*, pages 3–41. Elsevier.

Carlo Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 70–74.

EMOMINER at SemEval-2019 Task 3: A Stacked BiLSTM Architecture for Contextual Emotion Detection in Text

Nikhil Chakravartula
Teradata / Hyderabad
nikhil.chakravartula@gmail.com

Vijayasaradhi Indurthi
Teradata / Hyderabad
vijayasaradhi.indurthi@teradata.com

Abstract

This paper describes our participation in the SemEval 2019 Task 3 - Contextual Emotion Detection in Text. This task aims to identify emotions, viz. happiness, anger, sadness in the context of a text conversation. Our system is a stacked Bidirectional LSTM, equipped with attention on top of word embeddings pre-trained on a large collection of Twitter data. In this paper, apart from describing our official submission, we elucidate how different deep learning models respond to this task.

1 Introduction

Sentiment analysis is an established field in NLP, but just identifying positive and negative sentiments may not be enough. Applications require systems to go further beyond sentiment analysis and perform emotion analysis, which deals with identifying discrete emotions like anger, joy, sadness, etc. The task is challenging because the importance of context in emotion analysis cannot be overstated (Malik et al., 2017; Vanzo et al., 2014). Also, text in a conversation often contains language slangs, emoticons, emojis and other noisy data that make it difficult to identify the type of feeling expressed.

Many approaches have been put forward to identify emotions in a text. Purver and Battersby (2012); Balabantaray et al. (2012) used SVM classifier on twitter data to carry out emotion analysis. Potapova and Gordeev (2016) deployed a model based on Random Forests to identify aggression in texts. These approaches are often assisted by lexicons and require heavy feature engineering.

In recent years, deep learning approaches have outperformed traditional algorithms in NLP tasks (Bahdanau et al., 2014). Felbo et al. (2017); Gupta et al. (2017) utilized LSTMs to achieve emotion identification in text. Abdul-Mageed and Ungar (2017) showed that GRNNs achieved a very good performance on 24 fine-grained types of emotions. Kratzwald et al. (2018) proposed sent2affect, a tailored form of transfer learning for affective computing, where the network is pre-trained for sentiment analysis task, and subsequently the output layer is tuned to the task of emotion recognition. In this paper, we propose a stacked Bidirectional LSTM architecture to recognize emotions in text.

happy	sad	angry	others
14.06%	18.11%	18.25%	49.56%

Table 1: Train dataset composition.

The rest of the paper is organized as follows. In sec. 2 we give a brief explanation of the shared task. In sec. 3 we describe the process of feature engineering from the text. In sec. 4 we discuss system architecture. It is followed by sec. 5 which contains the various settings used in our experiments. In sec. 6, we analyse the results and conclude our paper in sec. 7 with future ideas and vision.

2 Shared Task Description

The SemEval 2019 Task 3 is as follows: Given three turns of a conversation by two users, say turn1 by user1, turn2 by user2 and turn3 by user1, the system must identify the emotion of turn3 based on the conversation. It has to be one of the four values - happy, sad, angry and others. The dataset is provided by the organizers of the task. The composition of the dataset is described in Table 1. More details about the task can be found in the task description paper (Chatterjee et al., 2019).

3 Feature Engineering

3.1 Pre Processing

We perform the following pre-processing operations on the text before feature engineering.

- All text is converted to lower case.
- All contractions are replaced with their full form. For example, *don't* will be replaced by *do not* and *can't* will be replaced by *can not*.
- All punctuation marks are removed.
- **Spell correction and Emoji expansion:** Many conversations include words in an elongated form (*nooooo, youuuuu, heyyyyyy etc.,*) and slangs(*wassup, 4u, lolz etc.,*). We perform spell corrections (Jurafsky and Martin, 2018) on these words to reduce the vocabulary size and to account for better results. Text8[1] is utilized to generate unigram and bigram word statistics with ekphrasis (Baziotis et al., 2017) to perform spell correction.

  Emojis play a crucial part in identifying the emotion of a conversation. A conversation often contains a high number of emojis that intrinsically determines its nature. Identifying this quintessential importance, we use a python package named emoji[2] to expand the emojis into representative keywords. Eg: 'unamused face'
- **Parts Of Speech**: Part-of-speech (POS) tagging is an important and fundamental step in Natural Language Processing. The Part-of-speech gives a large amount of information about a word and its neighbours, syntactic categories of words and similarities and dissimilarities between them. NLTK (Steven Bird and Loper, 2009) is used to extract the Parts Of Speech tags for each word in the conversation, and then concatenated them with GloVe vectors. As a result, Glove vectors will have syntactic information of words (Rezaeinia et al., 2017).

3.2 Feature Extraction

- **Word Embeddings:** Glove840B - common crawl (Pennington et al., 2014) pre-trained word embeddings are used to convert each of the words in the conversation to a 300-dimensional feature vector.
- **One Hot Encoding:** The POS tags generated in the previous step are converted to a constant vector using One-Hot Encoding
- **Lexicon:** We exploit the DepecheMood affective lexicon Deepechemood++ (Araque et al., 2018) that has been built in a completely unsupervised fashion, from affective scores assigned by readers to news articles. DepecheMood++ allows for both high-coverage and high-precision, providing scores for 187k entries on the following affective dimensions: Afraid, Happy, Angry, Sad, Inspired, Don't Care, Inspired, Amused, Annoyed.

4 System Architecture

4.1 Embedding Layer1 (EL1)

This embedding layer takes as input a fixed sequence of 200 words and converts each word into its corresponding 300 dimensional glove word vector (Pennington et al., 2014).

4.2 Embedding Layer2 (EL2)

This embedding layer takes as input a fixed sequence of 200 Parts Of Speech tags and converts each of them into a constant one-hot vector.

4.3 Embedding Layer3 (EL3)

This embedding layer takes as input a fixed sequence of 200 words and converts each of them into a vector based on the values in DepecheMood affective lexicon.

4.4 BiLSTM

Long Short-Term Memory (LSTM) is a recurrent neural network (RNN) architecture that has been designed to address the vanishing and exploding gradient problems of conventional RNNs. Unlike feed-forward neural networks, RNNs have cyclic connections making them powerful for modelling sequences. They have been successfully used for sequence labelling and sequence prediction tasks (Sak et al., 2014). An LSTM has 3 types of gates, the forget gate, the input gate and the output gate. The information flow is governed by the following equations.

[1] http://mattmahoney.net/dc/textdata.html

[2] https://github.com/carpedm20/emoji/

S.No	Setting	$F1\mu_{avg}$
1	EL1 + LSTM(256) + dropout(0.3)	0.8891
2	EL1 + BiLSTM(256) + dropout(0.2)	0.8903
3	EL1 + LSTM(256) + dropout(0.3) + Attention	0.8950
4	EL1 + BiLSTM(256) + dropout(0.3) + Attention	0.8918
5	EL1 + BiLSTM(128) + dropout(0.2) + BILSTM(128) + dropout(0.3)	0.8951
6	EL1 + BiLSTM(128) + dropout(0.2) + BILSTM(128) + dropout(0.3) + Attention	**0.8956**
7	EL1 + EL2 + BiLSTM(128) + dropout(0.2) + BiLSTM(128) + dropout(0.3)	0.8965
8	EL1 + EL3 + BiLSTM(128) + dropout(0.2) + BiLSTM(128) + dropout(0.3)	0.8969
9	EL1 + EL2 + EL3 + BiLSTM(128) + dropout(0.2) + BiLSTM(128) + dropout(0.3)	0.8931

Table 2: Results of different settings. S.No 1-6 are the variations of the system that are evaluated for the competition. S.No 7-9 provide further analysis of the system after the competition ended. All results shown are obtained with five fold cross validation on the train set.

$$\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) && (1)\\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) && (2)\\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) && (3)\\
C_t &= f_t \times C_{t-1} + i_t * \tilde{C}_t && (4)\\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) && (5)\\
h_t &= o_t \times \tanh C_t && (6)
\end{aligned}$$

Where:

- W_i, W_f, W_o, W_c : are the trained weights.
- b_i, b_f, b_o, b_c : are the trained biases
- σ: is the sigmoid function.
- x_t : is the input at time step t
- c_t : is the cell state at time t
- h_t : is the output at time step t

Single directional LSTM can only use the contextual information from the past. Bidirectional LSTM can use the contexts of the past as well as the future, generating two independent sequences of LSTM output vectors (Schuster and Paliwal, 1997). The output at each time step is the concatenation of two output vectors from both the directions, i.e.,

$$h_t = \overrightarrow{h_t} \oplus \overleftarrow{h_t}$$

4.5 Dropout

Dropout is a regularization technique in which units and their connections are randomly dropped from the neural network during training (Srivastava et al., 2014). This prevents units from co-adapting too much. Dropout of *p* sets *p* fraction of units to 0 at each update during training time. We employ dropout in our system to avoid overfitting.

4.6 Attention

Not all words in a sentence contribute to a sentiment. A neural network armed with an attention mechanism can actually understand how to disregard the noise and focus on what's relevant. This is especially effective in sequence tasks as the network can choose to remember only that context that's relevant (Zhang et al., 2018).

5 Experiments

5.1 Evaluation Metrics

Evaluation will be done by calculating micro averaged F1 score($F1\mu$) for the three emotion classes i.e. *Happy*, *Sad* and *Angry*. The *Others* class is ignored in the evaluation.

- $P\mu = (\sum TPi) \div (\sum (TPi + FPi))$ $\forall i \in Happy, Sad, Angry$
- $R\mu = (\sum TPi) \div (\sum (TPi + FNi))$ $\forall i \in Happy, Sad, Angry$
- $F1\mu = (2 \times P\mu \times R\mu) \div (P\mu + R\mu)$

TPi is the number of samples of class i which are correctly predicted, FNi and FPi are the counts of Type-I and Type-II errors respectively for the samples of class i. Please note that both the precision and recall are micro-averaged.

5.2 Methodology

All the experiments are developed using the Scikit-Learn (Pedregosa et al., 2011) machine learning library and keras deep learning library (Chollet et al., 2015) with Tensorflow backend (Abadi et al., 2015). We concatenate turn1, turn2 and turn3 using a separator 'eos', read as 'end of sentence'. Similarly, we concatenate the POS tags of all the turns as well using the same separator. Five-fold cross validation is used to evaluate our models. In all our experiments, the batch size is 200, the learning rate is 0.008 and the number of epochs is 10. The loss is categorical cross-entropy and the optimizer used is rmsprop. In all the settings, the activation for LSTM/BiLSTM is tanh and the last layer is a dense layer of 4 units with sigmoid activation. Attention layer, if employed, is used after the last LSTM/BiLSTM layer. All our code is publicly available in a Github repository.[3]

Table 2 shows the results of different variations of the system.

6 Results and Analysis

The results show that attention based models outperform their corresponding equivalents. It is interesting to see from S.No 1-4 that BiLSTM outperforms LSTM when no attention is used but in the presence of attention, LSTM performs better than BiLSTM. The two layer BiLSTM with attention in S.No 6 surmounted all the other variations during the competition. Hence, we submitted this model and achieved an F1μ score of 0.6939.

S.No 7-9 show our further analysis of the system when different embedding layers are merged. We see that concatenating any one of POS and DepecheMood to the word vectors increased the performance in S.No 7-8, but not by much. However, concatenation of word vectors, POS and DepecheMood decreased performance, as shown in S.No 9.

7 Conclusion and Future Work

In this paper, we described a stacked BiLSTM deep learning model to detect emotion in context. We used glove pre-trained embeddings to convert each word into its corresponding word vector and then passed it on to two layers of BiLSTM, applied attention mechanism and finally, passed the intermediate inputs on to a dense layer of 4 units with sigmoid activations. We also depicted the results of adding different features to the pre-trained word vectors. Inspired by the work of Rezaeinia et al. (2017), in the future, we would like to examine more lexicon combinations to analyze the performance of the system. We would also like to make the system deeper to scrutinize how it responds.

[3]https://git.io/fhFG4

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Muhammad Abdul-Mageed and Lyle Ungar. 2017. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 718–728. Association for Computational Linguistics.

Oscar Araque, Lorenzo Gatti, Jacopo Staiano, and Marco Guerini. 2018. Depechemood++: a bilingual emotion lexicon built through simple yet powerful techniques. *CoRR*, abs/1810.03660.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

R C Balabantaray, Iiit Bhubaneswar, Mudasir Mohammad, and Nibha Sharma. 2012. N.: Multi-class twitter emotion classification: A new approach. *International Journal of Applied Information Systems*, pages 48–53.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada. Association for Computational Linguistics.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

François Chollet et al. 2015. Keras. https://keras.io.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625. Association for Computational Linguistics.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations. *CoRR*, abs/1707.06996.

Daniel Jurafsky and James H. Martin. 2018. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall.

Bernhard Kratzwald, Suzana Ilic, Mathias Kraus, Stefan Feuerriegel, and Helmut Prendinger. 2018. Decision support with text-based emotion recognition: Deep learning for affective computing. *CoRR*, abs/1803.06397.

Mubasher H. Malik, Syed Ali Raza, and H.M. Shehzad Asif. 2017. Context based emotion analyzer for interactive agent. *International Journal of Advanced Computer Science and Applications*, 8(1).

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Rodmonga Potapova and Denis Gordeev. 2016. Detecting state of aggression in sentences using CNN. *CoRR*, abs/1604.06650.

Matthew Purver and Stuart Adam Battersby. 2012. Experimenting with distant supervision for emotion classification. In *EACL 2012*.

Seyed Mahdi Rezaeinia, Ali Ghodsi, and Rouhollah Rahmani. 2017. Improving the accuracy of pre-trained word embeddings for sentiment analysis. *CoRR*, abs/1711.08609.

Hasim Sak, Andrew W. Senior, and Françoise Beaufays. 2014. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *CoRR*, abs/1402.1128.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45:2673–2681.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Ewan Klein Steven Bird and Edward Loper. 2009. *Natural Language Processing with Python– Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, Inc.

Andrea Vanzo, Danilo Croce, and Roberto Basili. 2014. A context-based model for sentiment analysis in twitter. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2345–2354. Dublin City University and Association for Computational Linguistics.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis : A survey. *CoRR*, abs/1801.07883.

EmoSense at SemEval-2019 Task 3: Bidirectional LSTM Network for Contextual Emotion Detection in Textual Conversations

Sergey Smetanin
National Research University Higher School of Economics
Moscow, Russia
sismetanin@gmail.com

Abstract

In this paper, we describe a deep-learning system for emotion detection in textual conversations that participated in SemEval-2019 Task 3 “EmoContext”. We designed a specific architecture of bidirectional LSTM which allows not only to learn semantic and sentiment feature representation, but also to capture user-specific conversation features. To fine-tune word embeddings using distant supervision we additionally collected a significant amount of emotional texts. The system achieved 72.59% micro-average F_1 score for emotion classes on the test dataset, thereby significantly outperforming the officially-released baseline. Word embeddings and the source code were released for the research community.

1 Introduction

Emotion detection has emerged as a challenging research problem that can make some valuable contribution not only in basic spheres like medicine, sociology and phycology but also in more innovative areas such as human-computer interaction. Nowadays, people increasingly communicate using text messages with dialogue systems, for which it is crucial to provide emotionally aware responses to users. The SemEval-2019 Task 3 “EmoContext” is focused on the contextual emotion detection in textual conversation. In EmoContext, given a textual user utterance along with 2 turns of context in a conversation, we must classify whether the emotion of the next user utterance is “happy”, “sad”, “angry” or “others” (4-point scale). For a detailed description see (Chatterjee et al., 2019).

In this paper, we present bidirectional LSTM for contextual emotion detection in textual conversations that participated in SemEval-2019 Task 3 “EmoContext”. The proposed architecture aims to capture not only semantic and sentiment feature representation from the conversation turns, but also to capture user-specific conversation features. We avoided using traditional NLP features like sentiment lexicons and hand-crafted linguistic features by substituting them with word embeddings which were calculated automatically from the text corpora. Based on this paper, we make the following contributions[1] freely available for the research community:

- The source code of the deep-learning system for emotion detection.
- Word embeddings fine-tuned for emotional detection in short texts.

The rest of the article is organized as follows. Section 2 gives a brief overview of the related work. In section 3 we describe the proposed architecture of LSTM used in our system. Section 4 is focused on the texts pre-processing and training process. Section 5 lays emphasis on the different system architectures and approaches we have tried. In conclusion, the performance of our system and further ways of research are presented.

2 Related Work

In recent years deep learning techniques have captured the attention of researchers due to their ability to significantly outperform traditional methods in sentiment analysis task (Tang et al., 2015). This fact has also been confirmed by previous iterations of SemEval competition, where leading solutions used convolutional neural networks (CNN) and long short-term memory (LSTM) networks (Cliche, 2017; Baziotis et al., 2017, 2018) as well as transfer learning techniques (Duppada et al., 2018). However, limited research was focused

[1] https://github.com/sismetanin/emosense-semeval2019-task3-emocontext

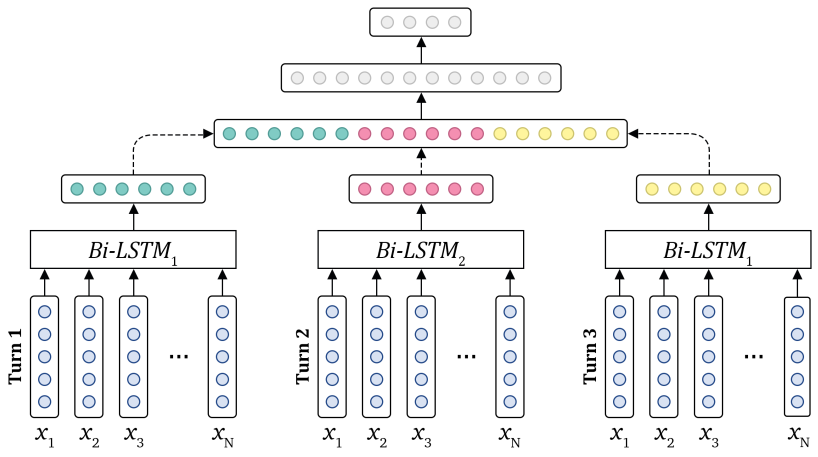


Figure 1: The architecture of a smaller version of the proposed architecture. LSTM unit for the first turn and for the third turn have shared weights.

on emotion identification in textual conversations. Since recurrent neural networks (RNNs) and their variations have been efficient in capturing sequential information, they have been successfully applied in emotion recognition systems (Poria et al., 2017; Gupta et al., 2017). Consequently, we draw our primary attention to the emotion classification in conversations using RRNs.

3 System Description

A recurrent neural network (RNN) is a family of artificial neural networks which is specialized in processing of sequential data. In contrast with traditional neural networks, RRNs are designed to deal with sequential data by sharing their internal weights processing the sequence. For this purpose, the computation graph of RRNs includes cycles, representing the influence of the previous information on the present one. As an extension of RNNs, Long Short-Term Memory networks (LSTMs) have been introduced in 1997 (Hochreiter and Schmidhuber, 1997). In LSTMs recurrent cells are connected in a special way in order to avoid vanishing and exploding gradient issues. Traditional LSTMs only preservs information from the past since they process the sequence only in one direction. Bidirectional LSTMs combine output from two hidden LSTM layers moving in opposite directions, where one moves forward through time, and another moves backwards through time, thereby enabling to capture information from both past and future states simultaneously (Schuster and Paliwal, 1997).

A high-level overview of our approach is provided in Figure 1. The proposed architecture of the neural network consists of the embedding unit and two bidirectional LSTM units $(dim = 64)$. The former LSTM unit is intended to analyze the utterance of the first user (i.e. the first turn and the third turn of the conversation), and the latter is intended to analyze the utterance of the second user (i.e. the second turn). These two units learn not only semantic and sentiment feature representation, but also how capture user-specific conversation features, which allows classifying emotions more accurately. At the first step, each user utterance is fed into corresponding bidirectional LSTM unit using pre-trained word embeddings. Next, these three feature maps are concatenated in a flatten feature vector and then passed to a fully connected hidden layer $(dim = 30)$, which analyzes interactions between obtained vectors. Finally, these features proceed through the output layer with the softmax activation function to predict a final class label. To reduce overfitting, regularization layers with Gaussian noise were added after the embedding layer, dropout layers (Srivastava et al., 2014) were added at each LSTM unit $(p = 0.2)$ and before the hidden fully connected layer $(p = 0.1)$.

4 Training

To train this model we had access to 30160 human-labelled tweets provided by task organizers, where about 5000 samples each from “angry”, “sad”, “happy” class and 15000 for “others” class (Table 1). Dev and test sets, which were also provided by organizers, in contrast with train set, have a real-life distribution, which is about 4% for each

Dataset	Happy	Sad	Angry	Others	Total
Train	14.07%	18.11%	18.26%	49.56%	30160
Dev	5.15%	4.54%	5.45%	84.86%	2755
Test	5.16%	4.54%	5.41%	84.90%	5509
Distant	33.3%	33.3%	33.3%	0%	900k

Table 1: Emotion class label distribution in datasets.

emotional class and the rest for the “others” class. Data provided by Microsoft.

In addition to this data, we collected 900k English tweets in order to create a distant dataset of 300k tweets for each emotion. To form the distant dataset, we based on the strategy of Go et al. (2009), under which we simply associate tweets with the presence of emotion-related words such as ’#angry’, ’#annoyed’, ’#happy’, ’#sad, ’#surprised’, etc. The list of query terms was based on the query terms of SemEval-2018 AIT DISC (Duppada et al., 2018).

The key performance metric of EmoContext is a micro-average F_1 score for three emotion classes, i.e. ‘sad’, ‘happy’, and ‘angry’. It is calculated as the harmonic mean of Precision and Recall.

4.1 Pre-processing

Before any training stage, texts were pre-processed by text pre-processing tool Ekphrasis (Baziotis et al., 2017). This tool helps to perform spell correction, word normalization and segmentation and allows to specify which tokens should be omitted, normalized or annotated with special tags. We used the following techniques for the pre-processing stage:

- URLs, emails, the date and time, usernames, percentage, currencies and numbers were replaced with the corresponding tags.
- Repeated, censored, elongated, and capitalized terms were annotated with the corresponding tags.
- Elongated words were automatically corrected based on built-in word statistics corpus.
- Hashtags and contractions unpacking (i.e. word segmentation) was performed based on built-in word statistics corpus.
- A manually created dictionary for replacing terms extracted from the text was used in order to reduce a variety of emotions.

In addition, Emphasis provides with the tokenizer which is able to identify most emojis, emoticons and complicated expressions such as censored, emphasized and elongated words as well as dates, times, currencies and acronyms.

4.2 Unsupervised Training

Word embeddings have become an essential part of any deep-learning approaches for NLP systems. To determine the most suitable vectors for emotions detection task, we try Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) and FastText (Joulin et al., 2017) models as well as DataStories pre-trained word vectors (Baziotis et al., 2017). The key concept of Word2Vec is to locate words, which share common contexts in the training corpus, in close proximity in vector space. Both Word2Vec and Glove models learn geometrical encodings of words from their co-occurrence information, but essentially the former is a predictive model and the latter is a count-based model. In other words, while Word2Vec tries to predict a target word (CBOW architecture) or a context (Skip-gram architecture), i.e. to minimize the loss function, GloVe calculates word vectors doing dimensionality reduction on the co-occurrence counts matrix. FastText is very similar to Word2Vec except for the fact that it uses character n-grams in order to learn word vectors, so it’s able to solve the out-of-vocabulary issue. For all techniques mentioned above, we used the default training prams provided by the authors. We train a simple LSTM model $(dim = 64)$ based on each of these embeddings and compare effectiveness using cross-validation. According to the result, DataStories pre-trained embeddings demonstrated the best average F_1 score.

4.3 Distant Pre-training

To enrich selected word embeddings with the emotional polarity of the words, we consider performing distant pre-training phrase by a fine-tuning of the embeddings on the automatically labelled distant dataset. The importance of using pre-training was demonstrated in (Deriu et al., 2017). We use the distant dataset to train the simple LSTM network to classify angry, sad and happy tweets. The embeddings layer was frozen for the first training epoch in order to avoid significant changes in the embeddings weights, and then it was unfrozen for the next 5 epochs. After the training stage, the fine-tuned embeddings was

System	Happy			Sad			Angry			Happy&Sad&Angry		
	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R
$Baseline$	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	58.61	n/a	n/a
$Random$	8.89	5.36	26.06	7.96	4.70	26.00	8.45	5.14	23.67	8.43	5.06	25.18
$LSTM_1$	67.07	59.07	77.68	76.95	71.57	83.36	71.18	61.75	84.67	71.37	63.31	81.89
$LSTM_2$	68.16	61.25	77.25	78.19	74.34	82.72	72.51	63.32	85.13	**72.58**	65.34	81.73
$LSTM_3$	67.33	60.70	75.77	75.23	69.60	82.00	70.06	59.08	86.27	70.58	62.34	81.41
$LSTM_s$	64.83	56.70	77.30	73.53	67.27	81.84	66.93	55.10	86.78	67.89	58.34	82.07
$LSTM_a$	66.50	58.71	77.00	75.64	71.40	80.71	69.88	59.27	85.56	70.30	62.15	81.19
$LSTM_w$	68.67	62.30	76.60	77.51	73.87	81.60	70.35	60.25	84.67	71.77	64.45	81.00

Table 2: Comparison of various models on dev dataset using micro-average Precision, Recall and F_1-score for emotional classes. $Baseline$ is an official baseline approach released by task organizers.

saved for the further training phases.

4.4 Supervised Training

At the final stage, the training dataset provided by SemEval-2019 was split into training and validation subsets. The validation subset was utilized as an unbiased accuracy evaluation of a model to fine-tune hyperparameters during training. The embedding layer was initialized with pre-trained word vectors from the previous distant training step. We use Adam optimizer (Kingma and Ba, 2014) with the initial learning rate of 0.001 and categorical cross-entropy as a loss function.

We train our network with frozen embeddings for the 15 epochs. We tried to unfrozen embeddings on the different epoch with the simultaneous reduction of learning rate but failed to get better results. It is probably connected with the size of the training dataset (Baziotis et al., 2017). The model was implemented using Keras with Tensorflow (Abadi et al., 2016) backend.

5 Experiments and Results

In the process of searching for optimal architecture, we experimented not only with the number of cells in layers, activation functions and regularization parameters but also with the architecture of the neural network. Let us take a closer look at the latter type of experiments. Comparison of various models presented in Table 2.

- $LSTM_1$ is a model with one bidirectional LSTM unit for all three conversation turns.
- $LSTM_2$ is a final model with two bidirectional LSTM units described in Section 2.
- $LSTM_3$ is a model with three bidirectional LSTM unit, where each unit is intended to analyze the corresponding conversation turn.
- $LSTM_w$ is $LSTM_2$ with an additional regularization based on class weights.
- $LSTM_s$ is $LSTM_2$ with an additional LSTM unit above concatenated layer.
- $LSTM_a$ is $LSTM_2$ with additional context-attention layer (Yang et al., 2016).

Since $LSTM_2$ demonstrated the best scores on the dev dataset, it was used in the final evaluation stage of the competition. On the final test dataset, it achieved 72.59% micro-average F_1 score for emotional classes. This is well above the official baseline released by task organizers, which was 58.68%.

6 Conclusion

In this paper, we presented the deep-learning system for emotion detection in textual conversations we used to compete in SemEval-2019 Task 3 ”EmoContext” competition. Utilizing state-of-the-art approaches in the literature, we decided to use RNNs to detect emotions. We designed a specific architecture of LSTM which allows not only to learn semantic and sentiment feature representation, but also to capture user-specific conversation features. In this work, we didn’t use any traditional NLP features such as sentiment lexicons or hand-crafted linguistic by substituting them with word embeddings which were calculated automatically from the text corpora with an advanced preprocessing stage.

Our approach achieved 72.59% micro-average F_1 score for emotion classes at the test dataset, thereby significantly outperform the officially-released baseline, namely larger in 14%. Further research will be focused on the advanced usage of techniques to handle imbalanced data. It also can be useful to consider the application of character-level language models.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, Berkeley, CA, USA. USENIX Association.

Christos Baziotis, Athanasiou Nikolaos, Pinelopi Papalampidi, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, and Alexandros Potamianos. 2018. Ntua-slp at semeval-2018 task 3: Tracking ironic tweets using ensembles of word and character level attentive rnns. In *Proceedings of The 12th International Workshop on Semantic Evaluation (SemEval-2018)*, pages 613–621. Association for Computational Linguistics.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754. Association for Computational Linguistics.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Mathieu Cliche. 2017. Bb_twtr at semeval-2017 task 4: Twitter sentiment analysis with cnns and lstms. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 573–580. Association for Computational Linguistics.

Jan Deriu, Aurelien Lucchi, Valeria De Luca, Aliaksei Severyn, Simon Müller, Mark Cieliebak, Thomas Hofmann, and Martin Jaggi. 2017. Leveraging large amounts of weakly supervised data for multi-language sentiment classification. In *Proceedings of the 26th international conference on world wide web*, pages 1045–1052. International World Wide Web Conferences Steering Committee.

Venkatesh Duppada, Royal Jain, and Sushant Hiray. 2018. Seernet at semeval-2018 task 1: Domain adaptation for affect in tweets. In *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018)*, pages 18–23. Association for Computational Linguistics.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations. *arXiv preprint arXiv:1707.06996*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, volume 2, pages 427–431. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, volume 2 of *NIPS'13*, pages 3111–3119, USA. Curran Associates Inc.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh, and Louis-Philippe Morency. 2017. Context-dependent sentiment analysis in user-generated videos. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 873–883. Association for Computational Linguistics.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Deep learning for sentiment analysis: Successful approaches and future challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(6):292–303.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489. Association for Computational Linguistics.

EPITA-ADAPT at SemEval-2019 Task 3: Using Deep Sentiment Analysis Models and Transfer Learning for Emotion Detection in Textual Conversations

Abdessalam Bouchekif[1] abdessalam.bouchekif@epita.fr
Praveen Joshi[2] praveen.joshi@mycit.ie
Latifa Bouchekif[3] latifa.bouchekif@univ-tlemcen.dz
Haithem Afli[2] haithem.afli@cit.ie

[1] LSE, EPITA Graduate School of Computer Science, France
[2] ADAPT Centre, Cork Institute of Technology, Cork, Ireland
[3] University Abou Bekr Belkaid-Tlemcen, Algeria

Abstract

Messaging platforms like WhatsApp, Facebook Messenger and Twitter have gained recently much popularity owing to their ability in connecting users in real-time. The content of these textual messages can be a useful resource for text mining to discover and unhide various aspects, including emotions. In this paper we present our submission for SemEval 2019 task 'EmoContext'. The task consists of classifying a given textual dialogue into one of four emotion classes : *Angry*, *Happy*, *Sad* and *Others*. Our proposed system is based on the combination of different deep neural networks techniques. In particular, we use Recurrent Neural Networks (LSTM, B-LSTM, GRU, B-GRU), Convolutional Neural Network (CNN) and Transfer Learning (TL) methods. Our final system, achieves an $F1\mu$ score of 74.51% on the subtask evaluation dataset.

1 Introduction

The world of text conversations has undergone drastic changes during the last few years. The generation and sharing of such information have become much easier than before with the advent of popular social media platforms such as Twitter, Facebook, *etc.* According to *Statistica*[1] WhatsApp is the most popular mobile messaging app in the world with one billion monthly active users. Facebook Messenger closely follows with 900 million monthly active users. The content of these messages can be useful resource for text mining to discover and unhide various aspects, including emotions (Chatterjee et al., 2019a).

Capturing and analysing these emotions from peoples conversations has raised growing interest within the scientific community in varied fields like cognitive and social psychology, signal processing and natural language processing (Gupta et al., 2017; Zhang et al., 2018; Majumder et al., 2018).

The goal of detecting emotions in SemEval2019 task3 described in (Chatterjee et al., 2019b) is to classify a given conversation into one of four classes - *happy*, *sad*, *angry* and *others*, that best represents the mental state of the users. This can be seen as a multiclass classification problem.

In this paper, we propose an approach to detect emotions like *happy*, *sad* or *angry* in textual messages using a combination of deep learning models. We apply, also, a transfer learning approach, from a model trained on a similar task consists on the prediction of the sentiment of the conversation, *i.e. positive*, *negative* or *neutral*. Then, the pretrained model is re-used to classify the dialogue into one of four classes : *happy*, *sad*, *angry* and *others*.

The rest of the paper is organized as follows. Section 2 provides a brief literature review on emotion detection in textual datasets. The description of our proposed system is presented in Section 3. The experimental set-up and results are described in Section 4. Finally, a conclusion is given with a discussion of future works in Section 5.

2 Related Work

Emotions are closely related to sentiment with more analysis of the inferred polarity. For example, a negative sentiment can be caused by sadness or anger, while a positive sentiment can be caused by happiness or anticipation. Thus, following the way in sentiment analysis, many deep learning models are applied to detect emotions (Zhang et al., 2018; Poria et al., 2017).

(Zhou et al., 2018) proposed an Emotional Chatting Machine (ECM) that can generate appropriate responses grammatically relevant and emotionally consistent based on GRU. Their system

1. `https://www.statista.com/topics/1145/internet-usage-worldwide/`

is modeling the emotion factor, using emotion category embedding, internal emotion memory, and external memory. A bilingual attention network model was proposed by (Wang et al., 2016) for code-switched emotion prediction. The authors used a LSTM model to construct a document level representation of each post, and an attention mechanism to capture the informative words from bilingual and monolingual contexts. (Abdul-Mageed and Ungar, 2017) built a large, automatically curated dataset for emotion detection using distant supervision and then used GRNNs to model fine-grained emotion. They extended the classification to model (Plutchik, 2001)'s 8 primary emotion dimensions.

(Felbo et al., 2017) show how millions of readily available emoji occurrences on Social Media can be used to pre-train models to learn a richer emotional representation. They transfer this knowledge to emotion, sarcasm and sentiment detection tasks using a new layer-wise fine-tuning method. In (Daval-Frerot et al., 2018) the authors used a transfer learning approach, from a model trained on a similar task. They propose to pre-train a model to predict if a tweet is positive, negative or neutral by applying a B-LSTM on an external dataset. Then, they used the pre-trained model to classify a tweet according to the seven-point (range from 5 to +5 respectively from very negative to very positive) scale of positive and negative sentiment intensity.

3 Proposed System

Figure 1 provides a high-level overview of our system, which consists of three steps :

1. First step applies the basic text processing (Tokenisation, lemmatisation, filtering the noise from the raw text data, *etc*) and represents words in textual dialogue as vectors.
2. In the second step, we learn a model for each one of the emotions : *angry*, *happy* and *sad*.
3. The last step does the prediction based on the probabilities of our three models. The system classifies the dialogue in one of four classes (*angry*, *happy*, *sad* and *others*).

In this work, we consider each conversation as one single input, *i.e.* we didn't take into account the writing turn (utterance). Our decision was based on the fact that the language and the size of these conversations are similar to the standard user generated content type of data sets.

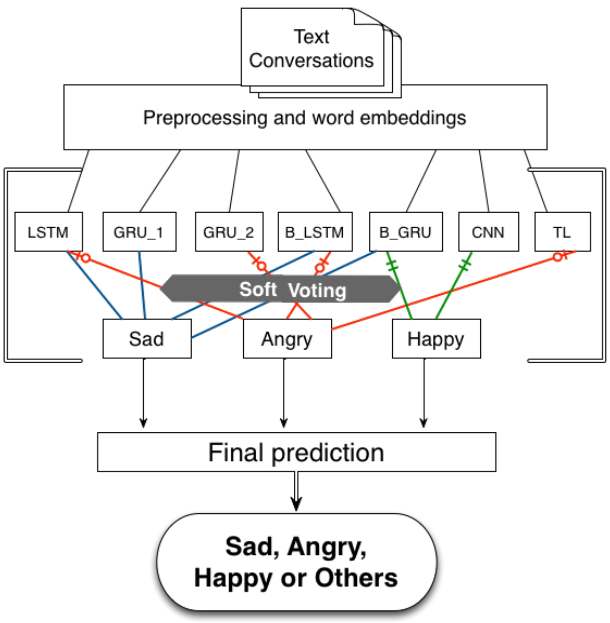


Figure 1 – Our proposed system architecture

3.1 Pre-processing and word representation

The textual dialogues are processed using *ekphrasis*[2] tool which allows performing the following tasks : tokenization, word normalization, word segmentation (for splitting hashtags) and spell correction (*i.e* replace a misspelled word with the most probable candidate word). All words are lowercased. E-mails, URLs and user handles are normalized. A detailed description of this tool is given in (Baziotis et al., 2017).
Each word in the text is represented by a vector of real numbers capturing the semantic meanings of words. We used datastories embeddings (Baziotis et al., 2017) trained on 330M english twitter messages posted from 12/2012 to 07/2016. The embeddings used in this work are 300 dimensional.

3.2 Neural Networks

With the recent advances in deep learning, the ability to analyse sentiments has considerably improved. Indeed, many experiments have used state-of-the-art systems to achieve high performance. For example, (Baziotis et al., 2017) use Bidirectional Long Short-Term Memory (B-LSTM) with attention mechanisms while (Deriu et al., 2016) use Convolutional Neural Networks (CNN). Both systems obtained the best performance at the the 2016 and 2017 SemEval 4-A task respecti-

2. https://github.com/cbaziotis/ekphrasis

vely. In this work, we use Long Short-Term Memory (LSTM), B-LSTM, Gated Recurrent Unit (GRU), Bidirectional-GRU (B-GRU) and CNN models and we, also, apply a Transfer Learning (TL) approach.

3.2.1 LSTM, GRU_1 and GRU_2 models

LSTM (Schuster and Paliwal, 1997) and GRU (Cho et al., 2014) are a special kind of RNN, capable of learning long-term dependencies. This ability comes from LSTM cells, that can decide which information to add and remove to the cell state regulated by gates (input gate, output gate and forget gate). The GRU cell is a simplified version of the LSTM cell.
LSTM and GRU_1 contain 2 layers of 128 neurons each. Similarly to (Baziotis et al., 2017), we added a Gaussian noise at the embedding layer with $\sigma = 0.3$ and dropout of 0.3 at LSTM/GRU layers. GRU_2 is similar to GRU_1, with some little differences where GRU_2 contains 3 layers of 100 neurons and dropout of 0.2 at the embedding layer.

3.2.2 B-LSTM and B-GRU models

The B-LSTM become a standard for deep sentiment analysis (Baziotis et al., 2017; Daval-Frerot et al., 2018; Moore and Rayson, 2017). B-LSTM and B-GRU consists of two LSTMs and two GRUs respectively in different directions running in parallel : the first forward network reads the input sequence from left to right and the second backward network reads the sequence from right to left. Each LSTM / GRU yields a hidden representation : $\vec{h}$ (left to right vector) and $\overleftarrow{h}$ (right-to-left vector) which are then combined to compute the output sequence. For our problem, capturing the context of words from both directions allows to better understand the textual conversations semantic. We use the same parameters of LSTM and GRU_1 models.

3.2.3 CNN model

The application of CNN models started with visual imagery. Many works apply CNN for sentiment analysis and obtained interesting results (Kim, 2014; Ouyang et al., 2015; Deriu et al., 2016). CNN is typically composed of three types of layers : convolution, pooling, and fully connected layers. Each neuron in the convolutional layer is connected only to a local region. In this work, we use multiple convolutional filters of sizes 3, 4 and 5.

3.2.4 TL-BLSTM model

In this work, we apply a TL, which allows to avoid learning from scratch. TL consists of transferring the knowledge learned on one task to a second related task. We start by training a first model to predict the sentiment of the dialogue : *positive*, *negative* or *neutral*. For this, we added a dense layer of 3 neurons to our B-LSTM (see subsection 3.2.2) . The model is learned using an external dataset[3] composed of 50333 tweets (7840 negatives, 19903 positives and 22590 neutrals). Then, the first model is re-used as the starting point to train a new model that classifies the dialogue into one of four classes : *happy*, *sad*, *angry* and *others*. For this, we remove the last layer of the pre-trained model and we add a fully-connected layer of 128 neurons followed by an output layer of 3 neurons (similar to our previous work (Daval-Frerot et al., 2018)).

4 Experimental settings and results

For each emotion, we use only the best models which maximizes the $F1\mu$ score. Table 2 presents the selected models for each emotion. Then, we combine these models using the soft voting approach considering only the probability of the selected emotion : *happy*, *sad* or *angry*.

For example, suppose that the CNN model gives 0.7, 0.1, 0.2 and 0.0 respectively for *happy*, *sad*, *angry* and *others*. And the GRU_1 model gives 0.6, 0.2, 0.0 and 0.2 for the same classes. We know, based on our results in table 2, that the CNN and GRU_1 are the best models for the emotion *happy*. So the soft voting combination is the average probability of 0.7 and 0.6, *i.e* 0.65.

In the last step we are applying a threshold after getting all prediction probabilities for our three classes. We choose threshold scores of 0.75 for angry and sad, and 0.67 for happy based on our experiments on the development set. These scores were high in order to avoid any possible confusion with the class *others*. This confusing can be caused by the fact that our training data is unbalanced.

Table 1 illustrates the performances of our system on *Dev* and *Test* sets. We can see an overall similarity in term of performance the system in both data sets. Indeed, the system achieves 74.4% and 74.5% on *Dev* and *test* sets respectively. The re-

3. `https://github.com/cbaziotis/datastories-semeval2017-task4/tree/master/dataset/Subtask_A/downloaded`

	Sad			Angry			Happy			Micro Average		
	P	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*	$P\mu$	$R\mu$	$F1\mu$
Dev	77.2	78.4	77.8	74.2	74.7	74.4	70.6	72.5	71.5	73.8	75.1	74.4
Test	82.5	75.6	78.9	73.9	75.8	74.8	70.8	70.1	70.4	75.2	73.8	74.5

Table 1 – performances of our system on *Dev* and *Test* corpora.

	models	$P\mu$	$R\mu$	$F1\mu$
Sad	B-GRU	73.7	79.6	76.5
	B-LSTM	72.6	78.8	75.6
	LSTM	80.1	71.2	75.4
	GRU_1	80.4	70.8	75.3
Angry	B-GRU	72.5	76.1	74.3
	GRU_2	69.5	78.1	73.6
	TL-BLSTM	70.1	76.5	73.1
	LSTM	72.2	71.4	71.8
Happy	CNN	69.4	68.6	69.6
	GRU_1	71.8	67.2	69.4

Table 2 – Models used for each emotion and their scores on test set.

sults shows a good performance for the *happy* and *Angry* and a better detection for the class *Sad* with a score around 78%.

Our final combination system, achieves an $F1\mu$ score of 74.51% on the test set gaining more than 3.5% compared to the best model before the combination which is B-GRU. This improvement comes from the fact that all systems used for the combination are using different methods that allows the diversification of their results and maximise the effect the soft voting.

Finally we can mention that after analysing our results, we have seen that most of errors came from the confusion between the class *Others* and the rest (*Happy*, *Angry* and *Sad*) which will be investigated in our future work.

5 Conclusion

In this paper, we propose to use a combination of different deep neural networks techniques including LSTM, B-LSTM, GRU, B-GRU, CNN and TL methods for the SemEval2019 task 3 of Emotions detection in textual conversations. Our system achieves a final $F1\mu$ of 74.51% on the subtask evaluation dataset.

As future work, we plan to develop an attention model to determine the importance of each part of the conversation (utterance) and its specific contribution to the emotion classification. We plan, also, to extend our work to other modalities such as audio emotions classification.

6 Acknowledgments

This research is partially supported by EPITA Systems Laboratory (www.lse.epita.fr) and Science Foundation Ireland through ADAPT Centre (Grant 13/RC/2106) (www.adaptcentre.ie).

References

Muhammad Abdul-Mageed and Lyle Ungar. 2017. Emonet : Fine-grained emotion detection with gated recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, page 718728, Vancouver, Canada.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 6 : Siamese LSTM with attention for humorous text comparison. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada.*

Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. 2019a. Understanding emotions in text using deep learning and big data. *Computers in Human Behavior*, 93 :309–317.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019b. Semeval-2019 task 3 : Emocontext : Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Guillaume Daval-Frerot, Abdessalam Bouchekif, and Anatole Moreau. 2018. Epita at semeval-2018 task 1 : Sentiment analysis using transfer learning approach. In *Proceedings of The 12th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT, New Orleans, Louisiana, June 5-6, 2018*, pages 151–155.

Jan Deriu, Maurice Gonzenbach, Fatih Uzdilli, Aurélien Lucchi, Valeria De Luca, and Martin Jaggi. 2016. Swisscheese at semeval-2016 task 4 : Sentiment classification using an ensemble of convolutional neural networks with distant supervision. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT, USA*.

Bjarke Felbo, Alan Mislove, Anders Sgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, page 16151625.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations. In *Proceedings of Neu-IR 2017 SIGIR Workshop on Neural Information Retrieval (Neu-IR 17)*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882.

Navonil Majumder, Soujanya Poria, Devamanyu Hazarika, Rada Mihalcea, Alexander Gelbukh, and Erik Cambria. 2018. Dialoguernn : An attentive rnn for emotion detection in conversations. In *arXiv :1811.00405v3*.

Andrew Moore and Paul Rayson. 2017. Lancaster A at semeval-2017 task 5 : Evaluation metrics matter : predicting sentiment from financial news headlines. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Canada*.

X. Ouyang, P. Zhou, C. H. Li, and L. Liu. 2015. Sentiment analysis using convolutional neural network. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 2359–2364.

Robert Plutchik. 2001. The nature of emotions human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice. In *American scientist*, volume 89(4), page 344350.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Mazumder, Amir Zadeh, and Louis-Philippe Morency. 2017. Context-dependent sentiment analysis in user-generated videos. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, page 873883, Vancouver, Canada.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*.

Zhongqing Wang, Yue Zhang, Sophia Yat Mei Lee, Shoushan Li, and Guodong Zhou. 2016. A bilingual attention network for code-switched emotion. In *Proceedings of the International Conference on Computational Linguistics (COLING 2016)*.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis : A survey. In *arXiv :1801.07883v2*.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018. Emotional chatting machine : Emotional conversation generation with internal and external memory. In *Proceedings of The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18))*.

Figure Eight at SemEval-2019 Task 3: Ensemble of Transfer Learning Methods for Contextual Emotion Detection

Joan Xiao
Figure Eight Inc.
San Francisco, CA 94103
`joan.xiao@gmail.com`

Abstract

This paper describes our transfer learning-based approach to contextual emotion detection as part of SemEval-2019 Task 3. We experiment with transfer learning using pre-trained language models (ULMFiT, OpenAI GPT, and BERT) and fine-tune them on this task. We also train a deep learning model from scratch using pre-trained word embeddings and BiLSTM architecture with attention mechanism. The ensembled model achieves competitive result, ranking ninth out of 165 teams. The result reveals that ULMFiT performs best due to its superior fine-tuning techniques. We propose improvements for future work.

1 Introduction

Traditionally sentiment analysis attempts to classify the polarity of a given text at the document, sentence, or feature/aspect level, i.e., whether the expressed opinion in the text is positive, negative, or neutral. More advanced sentiment classification looks at emotional states such as “Angry”, “Sad”, and “Happy”.

Due to the increasing popularity of social media, over the past years sentiment analysis tasks in SemEval competitions have been mostly focused on twitter (Rosenthal et al., 2014) (Rosenthal et al., 2015; Nakov et al., 2016; Rosenthal et al., 2017). SemEval-2018 Task 1: Affect in Tweets (Mohammad et al., 2018) includes an array of subtasks on inferring the emotions (such as joy, fear, valence, and arousal) of a person from his/her tweet.

As we increasingly communicate using text messaging applications and digital agents, contextual emotion detection in text is gaining importance to provide emotionally aware responses to users. SemEval-2019 Task 3 (Chatterjee et al., 2019) introduces a task to detect contextual emotion in conversational text.

Deep-learning based approaches have recently dominated the state-of-the-art in sentiment analysis. However, a good performing model often requires large amounts of labeled data and takes many days to train. In computer vision, transfer learning has enabled deep learning practitioners to leverage models that have been pre-trained on ImageNet, MS-COCO, and other large datasets (Razavian et al., 2014; Shelhamer et al., 2017; He et al., 2016; Huang et al., 2017). Fine-tuning such pre-trained models in computer vision has been a far more common practice than training from scratch.

In Natural Language Processing (NLP), the most common and simple transfer learning technique is fine-tuning pre-trained word embeddings (Mikolov et al., 2013). These embeddings are used as the first layer of the model on the new dataset, and still require training from scratch with large amounts of labeled data to obtain good performance.

In 2018 several pre-trained language models (ULMFiT, OpenAI GPT and BERT) emerged. These models are trained on very large corpus, and enable robust transfer learning for fine-tuning NLP tasks with little labeled data.

In SemEval-2019 Task 3, we apply transfer learning approach using both pre-trained word embeddings and pre-trained language models. Our model achieves highly competitive result.

In this paper we describe our approach and experiments. The rest of the paper is laid out as follows: Section 2 provides an overview of the task, Section 3 describes the system architecture, and Section 4 reports results and performs an error analysis to obtain a better understanding of strengths and weaknesses of our approach and subsequently proposes improvements. Finally

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 220–224
Minneapolis, Minnesota, USA, June 6–7, 2019. ©2019 Association for Computational Linguistics

Label	Train	Dev	Test
Happy	4,243	142	284
Sad	5,463	125	250
Angry	5,506	150	298
Others	14,948	2,338	4,677
Total	30,160	2,755	5,509

Table 1: Train/Dev/Test set in SemEval2019 Task 3.

we conclude in Section 5 along with a discussion about future work.

2 Task Overview

2.1 Dataset

The organizers provide a training, development, and test set. Each row in the dataset is a 3-turn conversation between two people. The task is to classify the emotion of a conversation as “Happy”, “Sad”, “Angry”, or “Others”. Table 1 shows the distribution of the datasets across the labels. No other dataset is used in our experiments.

2.2 Evaluation Metric

Evaluation metric is micro-averaged F1 score for the three emotion classes i.e. Happy, Sad and Angry (excluding the class “Others”). This is referred as micro F1 score throughout the paper.

3 System Description

3.1 System Architecture

Figure 1 details the System Architecture. We now describe how all the different modules are tied together. The input raw text is pre-processed as described in Section 3.2. The processed text is passed through all the models described in Sections 3.3 to 3.8. Finally, the system returns the average of the predicted probabilities from all models as the output.

3.2 Pre-processing

The conversation text in the dataset is similar to tweets in that it may contain one or many emojis, and may have misspelled words. We use ekphrasis tool [1] to preprocess the data. The tool performs the following steps: tokenization, spell correction (i.e replace a misspelled word with the most probable candidate word), word normalization, and word segmentation. All words are lower-cased.

[1]https://github.com/cbaziotis/ ekphrasis

After the text in each turn is processed, we concatenate them with a separator ”$\langle$eos$\rangle$”.

3.3 Fine-tuning ULMFiT

Universal Language Model Fine-tuning (ULMFiT) (Howard and Ruder, 2018) trains language models on Wikitext-103 (Merity et al., 2017b), which consists of 28,595 preprocessed Wikipedia articles and 103 million words. It’s based on the language model AWD-LSTM (Merity et al., 2017a), a regular LSTM (with no attention, shortcut connections, or other sophisticated additions) with various tuned dropout hyperparameters. It provides two pre-trained models: a forward model trained from left to right, a backward model trained from right to left.

Furthermore, it introduced several novel techniques for transfer learning: discriminative fine-tuning, slanted triangular learning rates, and gradual unfreezing, to retain previous knowledge and avoid catastrophic forgetting during fine-tuning.

We first fine-tune the forward language model. We combine all data including training, dev and test set, and split into a training and validation set. We use fast.ai’s lr_find [2] method to find the optimum learning rate, and use early stopping on validation loss to tune the dropout values from 0.7 to 2.5.

Then we fine-tune the classifier on the training set using 10-fold cross validation. We use early stopping on the evaluation metric of the task (micro F1 with ”Others” class excluded). We experiment with dropout values from 0.7 to 0.85.

We repeat the same process for the backward language model.

3.4 Fine-tuning BERT

Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) trains language models on BooksCorpus (800M words) (Zhu et al., 2015) and English Wikipedia (2,500M words). It trains a deep bidirectional language models by masking some percentage of the input tokens at random, and then predicting only those masked token. This creates deep bidirectional representations by jointly conditioning on both left and right context in all layers.

In addition, it also trains a binarized next sentence prediction task which helps with understanding relation between two sentences, important for

[2]https://github.com/fastai/fastai

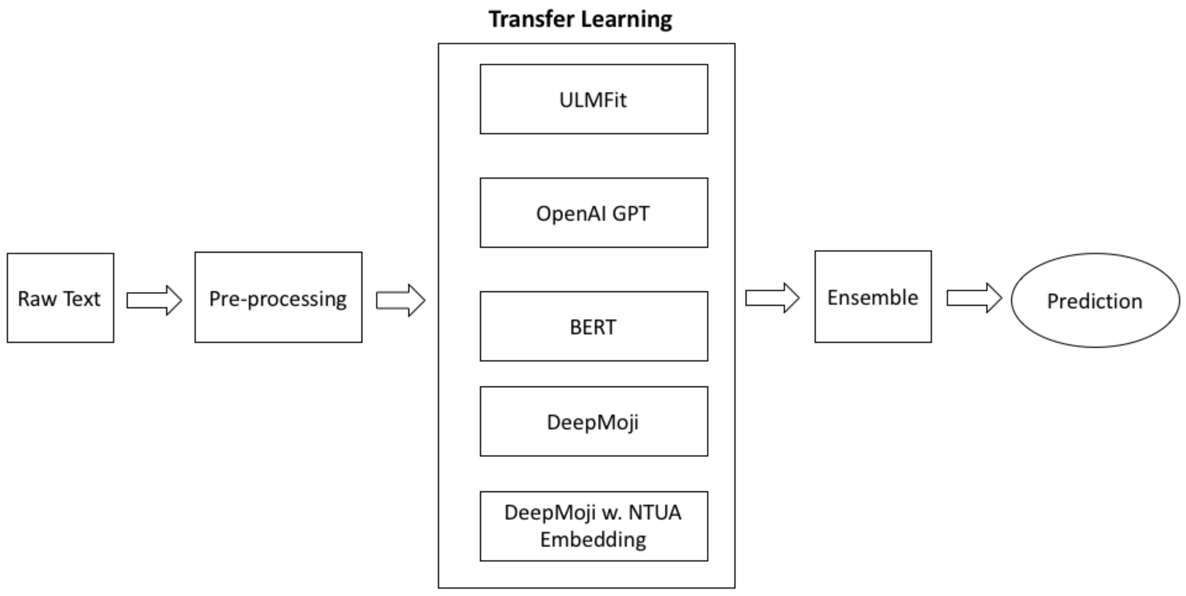


Figure 1: System Architecture.

Question Answering and Natural Language Inference tasks.

BERT provides pre-trained base and large models in multiple languages. In our experiments we use the large uncased English model. And we use the pytorch implementation by huggingface [3].

We experiment with fine-tuning the language model on a training and validation set split from a combined data set including training, dev and test set. We use early stopping on validation loss.

We then add a classifier layer on top of the output from the language model, and train it using the training set from the task with 10-fold cross validation. We use early stopping on the evaluation metric of the task (micro F1 with "Others" class excluded). We experiment with learning rate from 7e-6 to 3e-5.

3.5 Fine-tuning OpenAI GPT

OpenAI's Generative Pre-Training (GPT) (Alec et al., 2018) trains a language model using Transformer architecture on BooksCorpus. It obtains state-of-the-art result on many tasks including Natural Language Inference, Question answering and commonsense reasoning, Semantic Similarity, and Text Classification.

We tune the hyperparameters (clf_pdrop, embd_pdrop, resid_pdrop and attn_pdrop) in different combinations of values 0.1 and 0.2 (default value is 0.1) on the dev set. Due to that the dev score is less promising than the previous approaches, we do not use cross validation as it would take significant more time and compute resources. In fact this model was not included in the final submission.

3.6 Fine-tuning DeepMoji

DeepMoji (Felbo et al., 2017) performs distant supervision on a dataset of 1246 million tweets containing one of 64 common emojis. It obtained state-of-the-art performance on 8 benchmark datasets within sentiment, emotion and sarcasm detection using a single pretrained model.

We perform fine-tuning using the training set for training, and dev set as a validation set. We adopt the gradual unfreezing apporach (introduced by ULMFiT): first unfreeze the last layer and fine-tune all unfrozen layers for one epoch. We then unfreeze the next lower frozen layer and repeat, until we fine-tune all layers until convergence at the last iteration.

We do not use 10-fold cross validation due to that the highest micro F1 score on dev set does not seem promising.

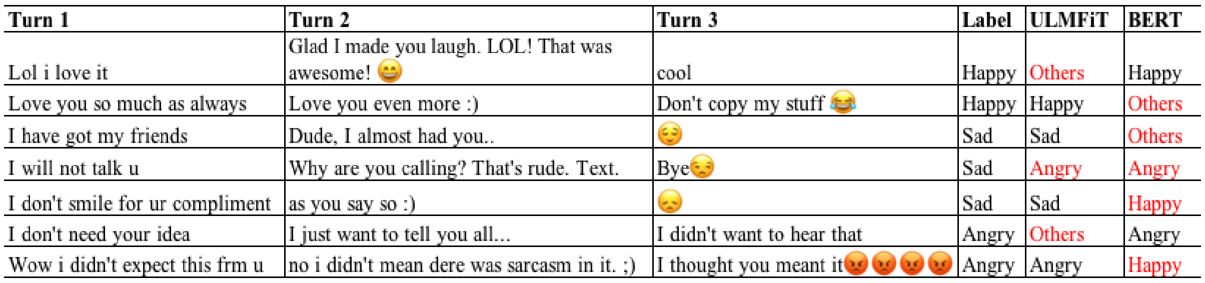
3.7 Training a DeepMoji model with NTUA embedding

We also train a model from scratch using the DeepMoji's architecutre, but replace its embedding with a 310 dimensional embedding trained by NTUA-SLP team (Baziotis et al., 2018), which was trained on a dataset of 550M English twit-

[3]https://github.com/huggingface/pytorch-pretrained-BERT

Model	F1 (Dev)				F1 (Test)			
	Happy	Sad	Angry	Avg	Happy	Sad	Angry	Avg
ULMFiT Fwd	0.7138	0.8106	0.7593	0.7586	0.6901	0.7647	0.7535	0.7357
ULMFiT Bwd	0.7101	0.8077	0.752	0.7541	0.6993	0.7598	0.7387	0.7321
BERT	0.6585	0.7574	0.7403	0.7172	0.6289	0.7040	0.7345	0.6907
OpenAI GPT	0.6322	0.7481	0.7395	0.7050	0.6388	0.7279	0.7280	0.6976
DeepMoji	0.6195	0.7037	0.7435	0.6914	0.5933	0.6932	0.7190	0.6703
DeepMoji/NTUA	0.7066	0.7881	0.7011	0.7274	0.6997	0.7518	0.7048	0.7168
Combined (all)	0.7097	0.8077	0.7656	0.7585	0.7267	0.8023	0.7776	0.7680
Combined (no OpenAI)*	0.7285	0.8244	0.7761	**0.7742**	0.7153	0.7977	0.7713	0.7608
ULMFiT+BERT+OpenAI	0.7255	0.7658	0.8185	0.7619	0.7254	0.8031	0.7799	**0.7686**

Table 2: Micro Average F1 scores on dev set and test set. Bold indicates the highest F1 score on each dataset among the ensembled models. Asterisk indicates our final submission: ensemble of all models except OpenAI.



Turn 1	Turn 2	Turn 3	Label	ULMFiT	BERT
Lol i love it	Glad I made you laugh. LOL! That was awesome! 😄	cool	Happy	Others	Happy
Love you so much as always	Love you even more :)	Don't copy my stuff 😂	Happy	Happy	Others
I have got my friends	Dude, I almost had you..	😌	Sad	Sad	Others
I will not talk u	Why are you calling? That's rude. Text.	Bye😒	Sad	Angry	Angry
I don't smile for ur compliment	as you say so :)	😞	Sad	Sad	Happy
I don't need your idea	I just want to tell you all...	I didn't want to hear that	Angry	Others	Angry
Wow i didn't expect this frm u	no i didn't mean dere was sarcasm in it. ;)	I thought you meant it😡😡😡😡	Angry	Angry	Happy

Figure 2: Prediction examples by ULMFiT and BERT. Red indicates incorrect prediction.

ter messages. It was trained based on word2vec and has 310 dimensional embeddings, consisting of 300 dim word2vec embeddings and 10 dim affective dimensions.

We use the keras_lr_finder [4] method to find the optimum starting learning rate (with the fastest decrease in training loss), and train the model on the training set using 10-fold cross validation and early stopping on the evaluation metric of micro F1 score.

3.8 Ensembling

We combine the predictions of all models above by taking the unweighted average of the posterior probabilities for these models, and the final prediction is the class with the largest averaged probability.

4 Results and Analysis

Table 2 shows the results of various models on the dev set and test set. ULMFiT has the best performance on both dev and test sets, outperforming all other pre-trained models. The DeepMoji model trained from scratch with NTUA embedding ranks the second.

Figure 2 shows some examples where the ULMFiT or BERT makes incorrect predictions for the same conversations. We observe that BERT often makes incorrect predictions when emojis are present in the text, while ULMFiT is more robust to emojis. This suggests that the high performance of ULMFiT is due to not only the large corpus on which the language model is pre-trained on, but also the superior fine-tuning methods, such as discriminative fine-tuning, slanted triangular learning rates, and gradual unfreezing.

5 Conclusion

In this paper we describe our methods for contextual emotion detection. We achieved very competitive results in SemEval-2019 Task 3 using an ensemble of Transfer Learning models. We demonstrate that with sophisticated fine-tuning techniques in ULMFiT, transfer learning using pre-trained language models yields the highest performance, outperforming models trained from scratch. For future work we plan to explore these techniques with OpenAI GPT and BERT as well.

[4] https://github.com/surmenok/keras_lr_finder

References

Radford Alec, Narasimhan Karthik, Salimans Tim, and Ilya Sutskever Openai. 2018. Improving Language Understanding by Generative Pre-Training. Technical report.

Christos Baziotis, Nikos Athanasiou, Alexandra Chronopoulou, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, Shrikanth Narayanan, and Alexandros Potamianos. 2018. NTUA-SLP at SemEval-2018 Task 1: Predicting Affective Content in Tweets with Deep Attentive RNNs and Transfer Learning.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Technical report.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. Technical report.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339. Association for Computational Linguistics.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2017. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017a. Regularizing and Optimizing LSTM Language Models. Technical report.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017b. Pointer Sentinel Mixture Models.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Saif M Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. SemEval-2018 Task 1: Affect in Tweets. Technical report.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. SemEval-2016 Task 4: Sentiment Analysis in Twitter. Technical report.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. CNN features off-the-shelf: an astounding baseline for recognition. *CoRR*, abs/1403.6382.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 Task 4: Sentiment Analysis in Twitter. Technical report.

Sara Rosenthal, Saif M Mohammad, Preslav Nakov, Alan Ritter, Svetlana Kiritchenko, and Veselin Stoyanov Facebook. 2015. SemEval-2015 Task 10: Sentiment Analysis in Twitter. Technical report.

Sara Rosenthal, Preslav Nakov, Alan Ritter, and Veselin Stoyanov. 2014. SemEval-2014 Task 9: Sentiment Analysis in Twitter. Technical report.

Evan Shelhamer, Jonathan Long, and Trevor Darrell. 2017. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):640–651.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *CoRR*, abs/1506.06724.

GenSMT at SemEval-2019 Task 3: Contextual Emotion Detection in tweets using multi task generic approach

Dumitru C. Bogdan
Faculty of Mathematics and Computer Science, University of Bucharest
Human Language Technologies Research Center, University of Bucharest
bogdan27182@gmail.com

Abstract

In this paper, we describe our participation in SemEval-2019 Task 3: EmoContext - A Shared Task on Contextual Emotion Detection in Text. We propose a three layer model with a generic, multi-purpose approach that without any task specific optimizations achieve competitive results (f1 score of 0.7096) in the EmoContext task. We describe our development strategy in detail along with an exposition of our results.

1 Introduction

In recent years, emotion detection in text has become more popular due to its vast potential applications in marketing, artificial intelligence, education, politics, psychology, human-computer interaction, etc. Social platforms like Twitter and Facebook, enabled access to huge amount of textual data facilitating both theoretical and experimental research.

Consequently, sentiment analysis and emotion detection have gained the interest of researchers in natural language processing and other fields. While sentiment analysis refers to classifying a subjective text as positive, neutral, or negative; emotion detection identify specific types of feelings such as anger, joy, fear, or sadness.

SemEval is the International Workshop on Semantic Evaluation that has evolved from SensEval. The purpose of this workshop is to evaluate semantic analysis systems. Task 3 (Chatterjee et al., 2019) in this workshop, presents the task of detecting contextual emotion in a given three turn conversation where every turn is a short phrase (or tweet).

Our proposed model factors a *generic* approach to sentiment and emotion detection in tweets thus being suitable for several tasks in 2019 SemEval competition like task 3, task 5 or task 6. By not exploiting to the fullest specificity of task 3 (being a three turn conversation) and by not including transfer learning from any similar task we assumed a lower accuracy, being confident that in a longer term this approach will be useful for a large class of emotion detection tasks in text.

We leverage the latest developments in natural language processing, like *deep contextualized word representation* and a specially build submodel (encoder) as we will see in section 4.

2 Previous work

Sentiment classification, the task of detecting whether a text is positive or negative, has a long history of research. From the moment when (Pak and Paroubek, 2010) constructed a corpus of tweets for this task research in this area, on this type of corpus, increased considerably. Classifying tweets according to sentiment has many applications in political science, social sciences, market research, and many others (Martínez-Cámara et al., 2012).

It was only natural to extend research in this area to include a more fine grained classification of text, more than positive or negative. As result emotion classification in tweets started to gain interest.

Emotion detection is part of the broader area of Affective Computing that has the goal to help machines detect and simulate emotions (Picard, 1995). Psychology offers us a number of theories about how to represent emotions while two are the most important and the most often used in existing approaches in sentiment analysis: emotional categories and emotional dimensions.

Emotional categories approaches are concentrated on model emotions based on distinct emotion classes. The categorical model assumes that emotion categories clearly separable, are discrete

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 225–229
Minneapolis, Minnesota, USA, June 6–7, 2019. ©2019 Association for Computational Linguistics

. Ekman presented a basic emotion model that fits categorical model approach. (Ekman, 2005) concluded that the six basic emotions are anger, disgust, fear, happiness, sadness and surprise.

Anger, sadness and happiness are the subject of the current task with one important addition: the absence of context which can add ambiguity making the task classification task much more complex, without access to facial expression and speech. Important amount of work has been done regarding to speech and facial emotion recognition however text based emotion detection systems still needs attraction of researchers (Sebea et al., 2005). Various methods where developed in the past like: *Keyword Spotting Technique*, *Lexical Affinity Method* or *Learning-based Methods*.

A similar research area is where emotions or more precisely emotion combinations plays a significant role is detection of optimism/pessimism in texts (Caragea et al., 2018).

Hopefully participants to this task will provide models and tools that will add value and will contribute to this field in a similar manner as previous SemEval tasks did (Mohammad et al., 2018).

3 Data preparation

Being a multi-task system, data preparation was also constrained to basic text preparation and tweets specific procedures. In effect we have treated all three sentences as a paragraph. We have concatenated turn1, turn2, turn3 separated by dot thus transforming the dataset in a list of pairs (label, text).

Tweets specific prepossessing involved sanitisation (links were replaced with *_url_* tag and user mentions were replaced with *_entity_* tag) and most important, we have translated emoji to text (e.g an emoji representing a face with tears of joy was expanded to "face with tears of joy" text). Emoji conversion is very important and based on intermediary tests it can greatly influence accuracy of models on datasets containing tweets.

To run our experiments, we used the dataset provided by the task organizers (Chatterjee et al., 2019) as follows. In pre-evaluation period we have trained our models on *training-set* and evaluated our model versions on *dev-set*. Dataset provided by the organizers contained 30160 entries and we have separated them into 24999 entries for *training-set* and remaining 5161 for *dev-set*. Table 1 provides more information about categories split in each set.

Data	Angry	Sad	Happy	Others	Total
All	5506	5463	4243	14948	30160
Train	4560	4504	3544	12391	24999
Dev	946	959	699	2557	5161

Table 1: Data split by sets and categories

4 Approach

Our selected model has a classical layered approach as outlined in figure 1.

The first layer consists of few embedders, that receives the input data as a text sequence and converts it into a specific representation.

Resulted transformation is next feed to the middle layer where few encoders individually transforms input vectors to a final representation of input texts.

Next, final representations are concatenated into a large vector and transmitted to the final layer which is a dense layer those output is transformed into a probability distribution consisting of k probabilities, where k is the number of classes depending on the selected SemEval task: $k = 4$ for task 3, $k = 2$ for task 5 and task 6.

Along with layered approach, a parallel text transformation was considered, meaning that the input text was transformed into three different internal representations using three different encoders.

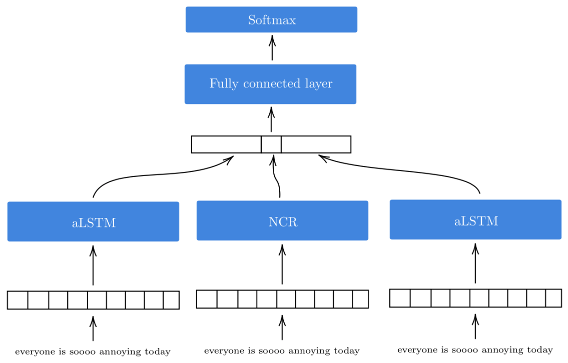

Softmax
Fully connected layer
aLSTM
NCR
aLSTM
everyone is soooo annoying today
everyone is soooo annoying today
everyone is soooo annoying today

Figure 1: GenSMT model structure

First encoder is ELMo (Peters et al., 2018), a deep contextualised word representation that models both complex characteristics of words syntax and semantics, and variations across linguistic contexts. Previous research and test showed significant improvements in various canonical NLP problems. A short description of this encoder is: ELMo word representations consists of functions of the entire input sentence computed on top of two-layer biLMs using character convolutions, as a linear function of the internal model states.

We can formalize this encoder as follows:

$$ELMo(x_{1:n}^{E_1}) = \gamma \sum_{j=0}^{l} s_j h_j^{LM}$$

where γ is a scalar parameter, l is the number of representation, s are softmax-normalized weights and h represents LSTM outputs.

Second encoder is NCR and works as follows: based on a list of words proposed by (Mohammad, 2018) it selects all occurrences of those words and their synonyms presented in the input text. Next it simply counts occurrences for each category (sad, anger, happy) and concatenates the results into a fixed size output vector.

We can formalize this encoder as follows:

$$NCR(x_{1:n}^{E_2}) = [\underset{cat \in \{anger, \dots\}}{count}(< cat >, x_{1:n})]$$

where $< cat >$ represents category and $count$ is the counting function.

Other NCR variations where proposed where we do not use a count function but we add, sub or dot product word representation and next we perform an operation of averaging, minimum or maximum.

Third encoder, RHN is an extended LSTM with recurrent dropout and possibility to use highway connections between internal layers.

We can formalize this encoder as follows:

$$RHN(x_{1:n}^{E_3}) = h \cdot t + x \cdot c$$

where h, t and c are internal compositions of nonlinear and linear functions.

Input data of the above encoders is formalized as follows:

$$\begin{aligned} x_{1:n}^{E_1} &= embedder_1(x_{1:n}) \\ x_{1:n}^{E_2} &= embedder_2(x_{1:n}) \\ x_{1:n}^{E_3} &= embedder_3(x_{1:n}) \end{aligned}$$

where $embedder_1$ produce character level representation of input text, $embedder_2$ and $embedder_3$ transform input words into their word embedding representation.

Now, we can formalize the full model as follows:

$$\begin{aligned} GenSMT(x_{1:n}) =& softmax(MLP(\Theta)); \\ \Theta =& [ELMo(x_{1:n}^{E_1}), \\ & NCR(x_{1:n}^{E_2}), \\ & RHN(x_{1:n}^{E_3})] \end{aligned}$$

As word representation we have use used various pretrained models in various configuration of GenSMT: Glove, Glove Twitter (Pennington et al., 2014), FastText (Joulin et al., 2016) and Wikipedia2vec (Yamada et al., 2018).

For development and testing we have used Python with ML related library like pytorch, numpy, pandas and others. Initial prototyping was done using Flair framework (Akbik et al., 2018) and development was done with AllenNLP (Gardner et al., 2017).

Training, validation and testing were performed on a single GPU machine. Training times were below one hour for each model configuration thus making this model suitable for hyper-parameters optimisation using grid search.

5 Experiments and results

As described in section 3 dataset was split in train and dev sets after being lightly pre-processed.

For training we use the model structure described in section 4 and we only changed model parameters and word embeddings of the embedders. We did not use any hyperparameters optimisation methods. Various model configurations were selected based on our previous experience with this kind of models.

Table 2 shows accuracy on dev tests of few selected configurations of GenSMT.

Model configuration	Accuracy
GenSMT$_1$	0.90
GenSMT$_2$	0.87
GenSMT$_3$	0.88
GenSMT$_4$	0.89

Table 2: Accuracy of various model configurations

$GenSMT_1$ configuration has the following general characteristics:

- $embeder_1$ performs character encoding
- $embeder_2$ uses 100d glove embeddings
- $embeder_2$ uses 500d wikipedia2vec embeddings
- ELMo large model
- MLP has three layers with ReLU activation
- batch size of 16
- Adam optimiser with a cosine learning rate scheduler
- 20 epochs with a patience of 5

Table 3 shows $GenSMT_1$ configuration where only default word embeddings of $embedder_3$ was changed. Results indicate that choosing word embeddings may influence accuracy.

Model configuration	Accuracy
GenSMT$_1(500d - wikipedia2vec)$	0.90
GenSMT$_1(300d - glove)$	0.89
GenSMT$_1(300d - fasttext)$	0.89

Table 3: Word-embeddings variations for a given configuration

We were interested to measure how each encoder contributes to model accuracy and for that ablation tests were performed. Table 4 shows $GenSMT_1$ configuration with one encoder removed.

Model configuration	Accuracy
GenSMT$_1(no\, ELMo)$	0.88
GenSMT$_2(no\, NCR)$	0.88
GenSMT$_3(no\, RHN)$	0.89

Table 4: Ablation tests on best performing configuration

While ELMo is powerful encoder, the gap between his absence and NCR absence, a simple encoder, is almost unnoticeable which is a surprise. This mean that a simpler model, with similar results, can be considered in scenarios where training or prediction speeds are of great importance (e.g mobile or IoT devices with low computing resources or low power resources).

For final, competition test we have used few $GenSMT_1$ configurations and the best performing one had an f1 score of 0.7096 placing this model in the upper half of the competition board.

6 Conclusions and future work

In this paper we have presented GenSMT model build as a generic system for SemEval 2019 task3, task 5 and task 6. It is a three layer model with three parallel embedders and encoders those outputs are concatenated and feed to dense layer which is next transformed into a probability distribution that produce text classification.

GenSMT was build with two constraint in mid. First we wanted to eliminate task specific dependencies as result we did not included any task specificity into the model. Second we did not use task specific transfer learning (e.g pretraining our model on a twitter sentiment dataset). This two constrains gave us the advantage to use the same model more than one task however reducing our accuracy capacity.

We have showed that choosing specific pretrained word embeddings can slightly improve the results, however greater gains are obtained by altering model hyperparameters.

By performing ablation tests we have showed that using a powerful encoder like ELMo increase the accuracy but not with an impressive score thus giving the option to use a simple and lower computational model to attain similar results much faster for both training and prediction.

Future work should explore two hypothesis. First as an upgrade of the model, it will be interesting to study how the model will perform if we remove the initial constraints and pretrain our model with a large twitter sentiment dataset followed by task specific data manipulation (e.g remove turn1 from dataset or inverse turns order). Second, as an update of the model, we can replace current encoder and embedders with new, state of the art ones.

Another aspect, specific for micro-blogging text is emoji presence and importance. While also present in larger texts, in short micro-blogging texts, emoji have more meaning, they carry a high quantity and quality of information that can be used for emotion detection thru data preprocessing or maybe using a model/encoder especially for them.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

Cornelia Caragea, Liviu P. Dinu, and Bogdan Dumitru. 2018. Exploring optimism and pessimism in twitter using deep learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 652–658. Association for Computational Linguistics.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Paul Ekman. 2005. Basic emotions. In *Handbook of Cognition and Emotion*, pages 45–60. John Wiley & Sons, Ltd.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Eugenio Martínez-Cámara, M. Teresa Martín-Valdivia, and L. Alfonso Ureña-López. 2012. Sentiment analysis in twitter. *Natural Language Engineering*, 20(01):1–28.

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 task 1: Affect in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 1–17. Association for Computational Linguistics.

Saif M. Mohammad. 2018. Word affect intensities. In *Proceedings of the 11th Edition of the Language Resources and Evaluation Conference (LREC-2018)*, Miyazaki, Japan.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. European Languages Resources Association (ELRA).

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

R. W. Picard. 1995. Affective computing.

Nicu Sebea, Ira Cohenb, and The Netherl. 2005. Multimodal approaches for emotion recognition: A survey.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2018. Wikipedia2vec: An optimized tool for learning embeddings of words and entities from wikipedia. *arXiv preprint 1812.06280*.

GWU NLP Lab at SemEval-2019 Task 3 : *EmoContext*: Effective Contextual Information in Models for Emotion Detection in Sentence-level in a Multigenre Corpus

Shabnam Tafreshi
George Washington University
Department of Computer Science
Washington, DC
shabnamt@gwu.edu

Mona Diab
AWS AI
George Washington University
Department of Computer Science
Washington, DC
mtdiab@gwu.edu

Abstract

In this paper we present an emotion classifier models that submitted to the SemEval-2019 Task 3 : *EmoContext*. The task objective is to classify emotion (*i.e.* happy, sad, angry) in a 3-turn conversational data set. We formulate the task as a classification problem and introduce a Gated Recurrent Neural Network (GRU) model with attention layer, which is bootstrapped with contextual information and trained with a multigenre corpus. We utilize different word embeddings to empirically select the most suited one to represent our features. We train the model with a multigenre emotion corpus to leverage using all available training sets to bootstrap the results. We achieved overall %56.05 f1-score and placed 144.

1 Introduction

In recent studies, deep learning models have achieved top performances in emotion detection and classification. Access to large amount of data has contributed to these high results. Numerous efforts have been dedicated to build emotion classification models, and successful results have been reported. In this work, we combine several popular emotional data sets in different genres, plus the one given for this task to train the emotion model we developed. We introduce a multigenre training mechanism, our intuition to combine different genres are a) to augment more training data, b) to generalize detection of emotion. We utilize Portable textual information such as subjectivity, sentiment, and presence of emotion words, because emotional sentences are subjective and affectual states like sentiment are strong indicator for presence of emotion.

The rest of this paper is structured as followings: section 2 introduce our neural net model, in section 3 we explain the experimental setup and data that is been used for training and development sets, section 4 discuss the results and analyze the errors, section 5 describe related works, section 6 conclude our study and discuss future direction.

2 Model Description

Gates Recurrent Neural Network (GRU) (Cho et al., 2014; Chung et al., 2015) and attention layer are used in sequential NLP problems and successful results are reported in different studies. Figure 1 shows the diagram of our model. [1]

GRU- has been widely used in the literature to model sequential problems. RNN applies the same set of weights recursively as follow:

$$h_t = f(W_{x_t} + U h_{t-1} + b) \tag{1}$$

GRU is very similar to LSTM with the following equations:

$$r_t = \sigma(W^r_{x_t} + U^r h_{t-1} + b^r) \tag{2}$$

$$z_t = \sigma(W^z_{x_t} + U^z h_{t-1} + b^z) \tag{3}$$

$$\hat{h_t} = \tanh(W_{x_t} + r_t \times U^{\hat{h}} h_{t-1} + b^{\hat{h}}) \tag{4}$$

$$h_t = z_t \times h_{t-1} + (1 - z_t) \times \hat{h_t} \tag{5}$$

GRU has two gates, a reset gate r_t, and an update gate z_t. Intuitively, the reset gate determines how to combine the new input with the previous memory, and the update gate defines how much of the previous memory to keep around. We use Keras[2] GRNN implementation to setup our experiments. We note that GRU units are a concatenation of GRU layers in each task.

Attention layer - GRUs update their hidden state h(t) as they process a sequence and the final

[1]Data and system will be released upon the request.
[2]https://keras.io/

hidden state holds the summation of all other history information. Attention layer (Bahdanau et al., 2014) modifies this process such that representation of each hidden state is an output in each GRU unit to analyze whether this is an important feature for prediction.

Model Architecture - our model has an embedding layer of 300 dimensions using *fasttext* embedding, and 1024 dimensions using ELMo (Peters et al., 2018) embedding. *GRU* layer has 70 hidden unites. We have 3 perceptron layers with size 300. Last layer is a *softmax* layer to predict emotion tags. Textual information layers (explained in section 2.1) are concatenated with GRU layer as auxiliary layer. We utilize a dropout (Graves et al., 2013) layer after the first perceptron layer for regularization.

2.1 Textual Information

Sentiment and objective Information (SOI)- relativity of subjectivity and sentiment with emotion are well studied in the literature. To craft these features we use SentiwordNet (Baccianella et al., 2010), we create sentiment and subjective score per word in each sentences. SentiwordNet is the result of the automatic annotation of all the synsets of WORDNET according to the notions of *positivity*, *negativity*, and *neutrality*. Each synset *s* in WORDNET is associated to three numerical scores Pos(s), Neg(s), and Obj(s) which indicate how positive, negative, and objective (i.e., neutral) the terms contained in the synset are. Different senses of the same term may thus have different opinion-related properties. These scores are presented per sentence and their lengths are equal to the length of each sentence. In case that the score is not available, we used a fixed score 0.001.

Emotion Lexicon feature (emo)- presence of emotion words is the first flag for a sentence to be emotional. We use NRC Emotion Lexicon (Mohammad and Turney, 2013) with 8 emotion tags (*e.i.* joy, trust, anticipation, surprise, anger, fear, sadness, disgust). We demonstrate the presence of emotion words as an 8 dimension feature, presenting all 8 emotion categories of the NRC lexicon. Each feature represent one emotion category, where 0.001 [3] indicates of absent of the emotion and 1 indicates the presence of the emotion. The advantage of this feature is their portability in transferring emotion learning across genres.

2.2 Word Embedding

Using different word embedding or end to end models where word representation learned from local context create different results in emotion detection. We noted that pre-trained word embeddings need to be tuned with local context during our experiments or it causes the model to not converge. We experimented with different word embedding methods such as *word2vec*, *GloVe* (Pennington et al., 2014), *fasttext* (Mikolov et al., 2018), and *ELMo*. Among these methods *fasttext* and *ELMo* create better results.

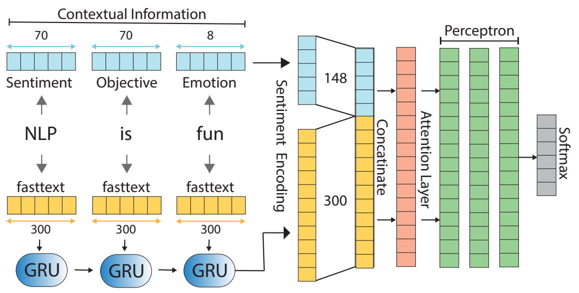


Figure 1: GRU-Attention neural net architecture. In this model framework, context information are features generated from SentiWordNet and emotion lexicon. We use fasttext to show the embedding layer (we use ELMo too, but we do not show it in here). Features are presented to GRU and attention layer and the output of attention layer is sent to 3 perceptron layer. Last layer is a softmax layer to predict emotion labels. Model without contextual info, exclude the contextual info input, which we do not show in the architecture.

3 Experimental Setup

We split *MULTI* dataset into 80%,10%,10% for train, dev, and test, respectively. We use AIT and EmoContext (data for this task) split as it is given by SemEval 2018 and semEval 2019. We describe these data sets in details in the next section. All experiments are implemented using Keras [4] and Tensorflow [5] in the back-end.

3.1 Data

We used three different emotion corpora in our experiments. Our corpora are as follows: a)

[3]empirically we observed that 0 is not a good initial value in neural net.

[4]https://keras.io/
[5]https://www.tensorflow.org/

A multigenre corpus created by (Tafreshi and Diab, 2018) with following genres: *emotional blog posts*, collected by (Aman and Szpakowicz, 2007), *headlines* data set from SemEval 2007-task 14 (Strapparava and Mihalcea, 2007), *movie review* data set (Pang and Lee, 2005) originally collected from Rotten tomatoes [6] for sentiment analysis and it is among the benchmark sets for this task. We refer to this multigenre set as (MULTI), b) SemEval-2018 Affect in Tweets data set (Mohammad et al., 2018) (AIT) with most popular emotion tags: *anger*, *fear*, *joy*, and *sadness*, c) the data set that is given for this task, which is 3-turn conversation data. From these data sets we only used the emotion tags *happy*, *sad*, and *angry*. We used tag *no-emotion* from MULTI data set as *others* tag. Data statistics are shown in figures 2, 3, 4 .

Data pre-processing - we tokenize all the data. For tweets we replace all the URLs, image URLs, hashtags, @users with specific anchors. Based on the popularity of each emoticon per each emotion tag, we replace them with the corresponding emotion tag. We normalized all the repeated characters, finally caps words are replaced with lower case but marked as caps words.

3.2 Training the Models

We have input size of 70 for sentence length, sentiment, and objective features and emotion lexicon feature has size 8. All these features are explained in section 2.1 and are concatenated with GRU layer as auxiliary (input) layer. Attention comes next after GRU and have size 70. We select dropout of size 0.2. We select 30 epochs in each experiment, however, training is stopped earlier if 2 consecutive larger loss values are seen on evaluation of dev set. We use Adam (Kingma and Ba, 2014) optimizer with a learning rate 0.001. We use dropout with rates 0.2. The loss function is a categorical-cross-entropy function. We use a mini batch (Cotter et al., 2011) of size 32. All hyper-parameter values are selected empirically. We run each experiment 5 times with random initialization and report the mean score over these 5 runs. In section 4 we describe how we choose the hyper-parameters values.

[6]https://www.rottentomatoes.com/

Data set	#train	#dev	#test	total
MULTI	13776	1722	1722	17220
AIT	6839	887	4072	11798
EmoContext	30160	2755	5510	38425

Table 1: Data statistics illustrating the distributions of the train, dev, and test sets across different data sets.

baseline- in each sentence we tagged every emotional word using NRC emotion lexicon (Mohammad and Turney, 2013), if any emotion has majority occurrence we pick that emotion tag as sentence emotion tag, when all emotion tags happen only once we randomly choose among them, when there is no emotional word we tag the sentence as *others*. We only use the portion of the emotion lexicon that covers the tags in the task (*i.e.* happy, sad, and angry).

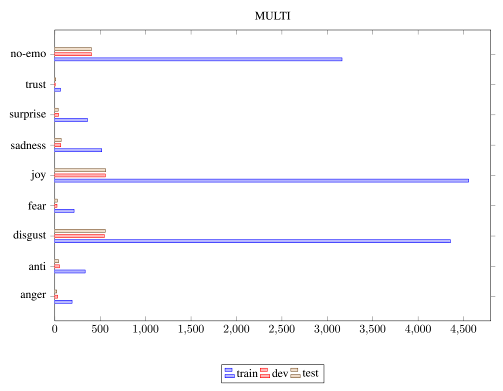


Figure 2: MULTI data set - train, dev, test data statistic

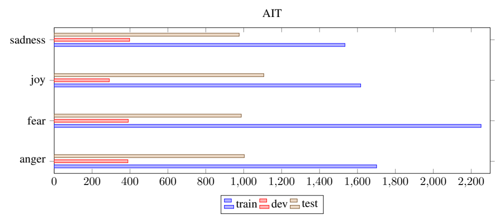


Figure 3: AIT data set - train, dev, test data statistic

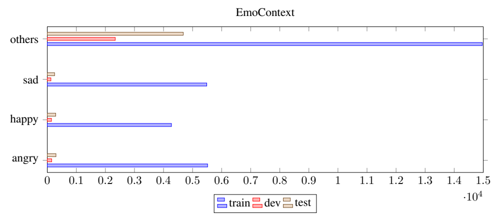


Figure 4: EmoContext data set - train, dev, test data statistic

Methods/Data set	**EmoContext**				
	pr.	re.	f.	acc.	sp./#epo.
Baseline	-	-	46.20	46.20	n.a.
GRU-att-fasttext	88.12	81.24	83.44	80.84	103/14
GRU-att-fasttext+F	88.27	84.47	85.27	82.07	321/8
GRU-att-ELMo	88.50	82.65	83.05	82.65	310/20
GRU-att-ELMo+F	**88.61**	**84.34**	**85.54**	**83.62**	960/28
Context Results(emotion only)	54.28	57.93	56.04	-	960/28

Table 2: Results on the *EmoContext* test sets. We report the mean score over 5 runs. Standard deviations in score are around 0.8. The experiments are demonstrating different embedding (*i.e.* ELMo and fasttext), with features (F), which are *emo* and *SOI* explained in section 2.1

Emotion tags/Data set	**EmoContext**		
	pr.	re.	f.
happy	45.37	53.52	49.11
sad	57.92	55.60	56.73
angry	61.02	64.09	62.52

Table 3: Context results of each emotion tag.

4 Results and Analysis

The results indicates the impact of contextual information using different embeddings, which are different in feature representation. Results of class *happy* without contextual features has %44.16 by GRU-att-ELMo model, and %49.38 by GRU-att-ELMo+F.

We achieved the best results combining ELMo with contextual information, and achieve %85.54 f-score overall, including class *others*. In this task we achieved %56.04 f-score overall for emotion classes, which indicates our model needs to improve the identification of emotion. Table 3 shows our model performance on each emotion tag. The results show a low performance of the model for emotion tag *happy*, which is due to our data being out of domain. Most of the confusion and errors are happened among the emotion categories, which suggest further investigation and improvement. We achieved %90.48, %60.10, %60.19, %49.38 f-score for class *others*, *angry*, *sad*, and *happy* respectfully.

Processing ELMo and attention is computationally very expensive, among our models GRU-att-ELMo+F has the longest training time and GRU-att-fasttext has the fastest training time. Results are shown in table 2 and table refemoresultss

5 Related Works

In semEval 2018 task-1, *Affect in Tweets* (Mohammad et al., 2018), 6 team reported results on sub-task E-c (emotion classification), mainly using neural net architectures, features and resources, and emotion lexicons. Among these works (Baziotis et al., 2018) proposed a Bi-LSTM architecture equipped with a multi-layer self attention mechanism, (Meisheri and Dey, 2018) their model learned the representation of each tweet using mixture of different embedding. in *WASSA 2017 Shared Task on Emotion Intensity* (Mohammad and Bravo-Marquez, 2017), among the proposed approaches, we can recognize teams who used different word embeddings: GloVe or word2vec (He et al., 2017; Duppada and Hiray, 2017) and exploit a neural net architecture such as LSTM (Goel et al., 2017; Akhtar et al., 2017), LSTM-CNN combinations (Köper et al., 2017; Zhang et al., 2017) and bi-directional versions (He et al., 2017) to predict emotion intensity. Similar approach is developed by (Gupta et al., 2017) using sentiment and LSTM architecture. Proper word embedding for emotion task is key, choosing the most efficient distance between vectors is crucial, the following studies explore solution sparsity related properties possibly including uniqueness (Shen and Mousavi, 2018; Mousavi and Shen, 2017) .

6 Conclusion and Future Direction

We combined several data sets with different annotation scheme and different genres and train an emotional deep model to classify emotion. Our results indicate that semantic and syntactic contextual features are beneficial to complex and state-of-the-art deep models for emotion detection and classification. We show that our model is able to classify non-emotion (others) with high accuracy. In future we want to improve our model to be able to distinguish between emotion classes in a more sufficient way. It is possible that hierarchical bi-directional GRU model can be beneficial, since these models compute history and future sequence while training the model.

References

Md S Akhtar, Palaash Sawant, Asif Ekbal, JD Pawar, and Pushpak Bhattacharyya. 2017. Iitp at emoint-2017: Measuring intensity of emotions using sentence embeddings and optimized features. In *Association for Computational Linguistics*.

Saima Aman and Stan Szpakowicz. 2007. Identifying expressions of emotion in text. In *International*

Conference on Text, Speech and Dialogue, pages 196–205. Springer.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *Lrec*, pages 2200–2204.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Christos Baziotis, Nikos Athanasiou, Alexandra Chronopoulou, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, Shrikanth Narayanan, and Alexandros Potamianos. 2018. Ntua-slp at semeval-2018 task 1: Predicting affective content in tweets with deep attentive rnns and transfer learning. *arXiv preprint arXiv:1804.06658*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, pages 2067–2075.

Andrew Cotter, Ohad Shamir, Nati Srebro, and Karthik Sridharan. 2011. Better mini-batch algorithms via accelerated gradient methods. In *Advances in neural information processing systems*, pages 1647–1655.

Venkatesh Duppada and Sushant Hiray. 2017. Seernet at emoint-2017: Tweet emotion intensity estimator. *arXiv preprint arXiv:1708.06185*.

Pranav Goel, Devang Kulshreshtha, Prayas Jain, and Kaushal Kumar Shukla. 2017. Prayas at emoint 2017: An ensemble of deep neural architectures for emotion intensity prediction in tweets. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 58–65.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations. *arXiv preprint arXiv:1707.06996*.

Yuanye He, Liang-Chih Yu, K Robert Lai, and Weiyi Liu. 2017. Yzu-nlp at emoint-2017: Determining emotion intensity using a bi-directional lstm-cnn model. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 238–242.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Maximilian Köper, Evgeny Kim, and Roman Klinger. 2017. Ims at emoint-2017: emotion intensity prediction with affective norms, automatically extended resources and deep learning. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 50–57.

Hardik Meisheri and Lipika Dey. 2018. Tcs research at semeval-2018 task 1: Learning robust representations using multi-attention architecture. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 291–299.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 task 1: Affect in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 1–17.

Saif M Mohammad and Felipe Bravo-Marquez. 2017. Wassa-2017 shared task on emotion intensity. *arXiv preprint arXiv:1708.03700*.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a word-emotion association lexicon. *Computational Intelligence*, pages 436–465.

Seyedahmad Mousavi and Jinglai Shen. 2017. Solution uniqueness of convex piecewise affine functions based optimization with applications to constrained

\

ell_1 $minimization. arXiv preprint arXiv:1711.05882.$

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Jinglai Shen and Seyedahmad Mousavi. 2018. Least sparsity of p-norm based optimization problems with p¿1. *SIAM Journal on Optimization*, 28(3):2721–2751.

Carlo Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *Proceedings of the 4th international workshop on semantic evaluations*, pages 70–74. Association for Computational Linguistics.

Shabnam Tafreshi and Mona T Diab. 2018. Sentence and clause level emotion annotation, detection, and classification in a multi-genre corpus. In *LREC*.

You Zhang, Hang Yuan, Jin Wang, and Xuejie Zhang. 2017. Ynu-hpcc at emoint-2017: Using a cnn-lstm model for sentiment intensity prediction. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 200–204.

IIT Gandhinagar at SemEval-2019 Task 3: Contextual Emotion Detection Using Deep Learning

Arik Pamnani*, **Rajat Goel***, **Jayesh Choudhari**, **Mayank Singh**
IIT Gandhinagar
Gujarat, India
{arik.pamnani,rajat.goel,choudhari.jayesh,singh.mayank}@iitgn.ac.in

Abstract

Recent advancements in Internet and Mobile infrastructure have resulted in the development of faster and efficient platforms of communication. These platforms include speech, facial and text-based conversational mediums. Majority of these are text-based messaging platforms. Development of *Chatbots* that automatically understand latent emotions in the textual message is a challenging task. In this paper, we present an automatic emotion detection system that aims to detect the emotion of a person textually conversing with a chatbot. We explore deep learning techniques such as CNN and LSTM based neural networks and outperformed the baseline score by 14%. The trained model and code are kept in public domain.

1 Introduction

In recent times, text has become a preferred mode of communication (Reporter, 2012; ORTUTAY, 2018) over phone/video calling or face-to-face communication. New challenges and opportunities accompany this change. Identifying sentiment from text has become a sought after research topic. Applications include detecting depression (Wang et al., 2013) or teaching empathy to chatbots (WILSON, 2016). These applications leverage NLP for extracting sentiments from text. On this line, SemEval Task 3: EmoContext (Chatterjee et al., 2019) challenges participants to identify Contextual Emotions in Text.

Challenges: The challenges with extracting sentiments from text are not only limited to the use of slang, sarcasm or multiple languages in a sentence. There is also a challenge which is presented by the use of non-standard acronyms specific to individuals and others which are present in the task's dataset 2.

*Equal Contribution

Existing work: For sentiment analysis, most of the previous year's submissions focused on neural networks (Nakov et al., 2016). Teams experimented with Recurrent Neural Network (RNN) (Yadav, 2016) as well as Convolutional Neural Network (CNN) based models (Ruder et al., 2016). However, some top ranking teams also used (Giorgis et al., 2016) classic machine learning models. Aiming for the best system, we started with classical machine learning algorithms like Support Vector Machine (SVM) and Logistic Regression (LR). Based on the findings from them we moved to complex models using Long Short-Term Memory (LSTM), and finally, we experimented with CNN in search for the right system.

Our contribution: In this paper, we present models to extract emotions from text. All our models are trained using only the dataset provided by EmoContext organizers. The evaluation metric set by the organizers is micro F1 score (**referred as score in rest of the paper**) on three {*Happy, Sad, Angry*} out of the four labels. We experimented by using simpler models like SVM, and Logistic regression but the score on dev set was below 0.45. We then worked with a CNN model and two LSTM based models where we were able to beat the baseline and achieve a maximum score of 0.667 on test set.

Outline: Section 2 describes the dataset and preprocessing steps. Section 3 presents model description and system information. In the next section (Section 4), we discuss experimental results and comparisons against state-of-the-art baselines. Towards the end, Section 5 and Section 6 conclude this work with current limitations and proposal for future extension.

2 Dataset

We used the dataset provided by Task 3 in SemEval 2019. This task is titled as *'EmoContext: Contextual Emotion Detection in Text'*. The dataset consists of textual dialogues i.e. a user utterance along with two turns of context. Each dialogue is labelled into several emotion classes: *Happy, Sad, Angry* or *Others*. Figure 1 shows an example dialogue.

Turn 1: N u
Turn 2: Im fine, and you?
Turn 3: I am fabulous 😎

Figure 1: Example textual dialogue from EmoContext dataset.

Table 1 shows the distribution of classes in the EmoContext dataset. The dataset is further subdivided into train, dev and test sets. In this work, we use training set for model training and dev set for validation and hyper-parameter tuning.

	Others	**Happy**	**Sad**	**Angry**
Train	14948	4243	5463	5506
Dev	2338	142	125	150
Test	4677	284	250	298

Table 1: Dataset statistics.

Preprocessing: We leverage two pretrained word embedding: (i) GloVe (Pennington et al., 2014) and (ii) sentiment specific word embedding (SSWE) (Tang et al., 2014). However, several classes of words are not present in these embeddings. We list these classes below:

- **Emojis:** 😎, 😇, etc.
- **Elongated words:** Wowwww, noooo, etc.
- **Misspelled words:** ofcorse, activ, doin, etc.
- **Non-English words:** Chalo, kyun, etc.

We follow a standard preprocessing pipeline to address the above limitations. Figure 2 describes the dataset preprocessing pipeline.

By using the dataset preprocessing pipeline, we reduced the number of words not found in GloVe embedding from *4154* to *813* and in SSWE from *3188* to *1089*.

[†]NLTK Library (Bird et al., 2009)

[‡]For spell check we the used the following PyPI package - pypi.org/project/pyspellchecker/.

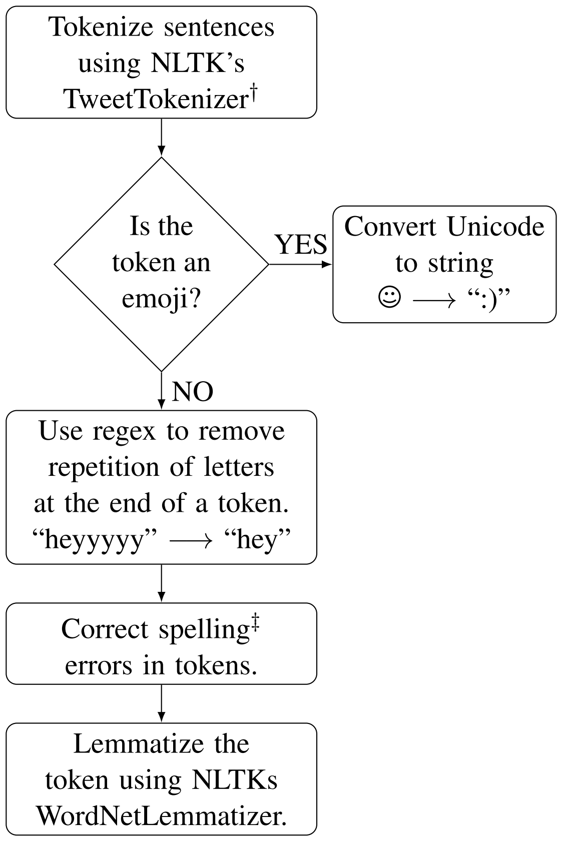

Tokenize sentences using NLTK's TweetTokenizer†
Is the token an emoji?
YES
Convert Unicode to string ☺ ⟶ ":)"
NO
Use regex to remove repetition of letters at the end of a token. "heyyyyy" ⟶ "hey"
Correct spelling‡ errors in tokens.
Lemmatize the token using NLTKs WordNetLemmatizer.

Figure 2: Data processing pipeline.

3 Experiments

We experiment with several classification systems. In the following subsections we first explain the classical ML based models followed by Deep Neural Network based models.

3.1 Classical Machine Learning Methods

We learn[§] two classical ML models, (i) Support Vector Machine (SVM) and (ii) Logistic Regression (LR). The input consists a single feature vector formed by combining all sentences (turns). We term this combination of sentences as *'Dialogue'*.

We create feature vector by averaging over d dimensional GloVe representations of all the words present in the dialogue. Apart from standard averaging, we also experimented with tf-idf weighted averaging. The dialogue vector construction from tf-idf averaging scheme is described below:

$$Vector_{dialogue} = \frac{\Sigma_{i=1}^{N}(\text{tf-idf}_{w_i} \times GloVe_{w_i})}{N}$$

[§]We leverage the Scikit-learn (Pedregosa et al., 2011) implementation.

Here, N is the total number of tokens in a sentence and w_i is the i^{th} token in the dialogue. Empirically, we found that, standard averaging shows better prediction accuracy than tf-idf weighted averaging.

3.2 Deep Neural Networks

In this subsection, we describe three deep neural architectures that provide better prediction accuracy than classical ML models.

3.2.1 Convolution Neural Network (CNN)

We explore a CNN model analogous to (Kim, 2014) for classification. Figure 3 describes our CNN architecture. The model consists of an embedding layer, two convolution layers, a max pooling layer, a hidden layer, and a softmax layer. For each dialogue, the input to this model is a sequence of token indices. Input sequences are padded with zeros so that each sequence has equal length n.

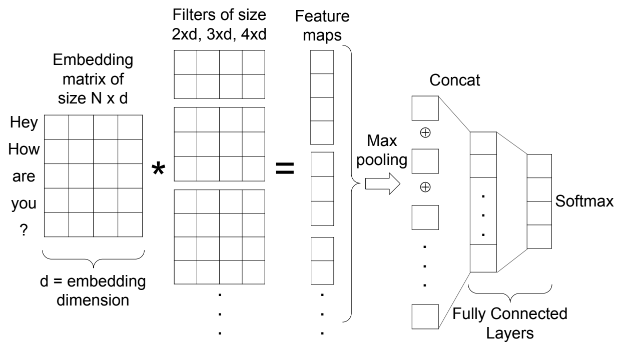


Figure 3: Architecture of the CNN model.

The embedding layer maps the input sequence to a matrix of shape $n \times d$, where n represents n^{th} word in the dialogue and d represents dimensions of the embedding. Rows of the matrix correspond to the GloVe embedding of corresponding words in the sequence. A zero vector represents words which are not present in the embedding.

At the convolution layer, filters of shape $m \times d$ slide over the input matrix to create feature maps of length $n - m + 1$. Here, m is the 'region size'. For each region size, we use k filters. Thus, the total number of feature maps is $m \times k$. We use two convolution layers, one after the other.

Next, we apply a max-pooling operation over each feature map to get a vector of length $m \times k$. At the end, we add a hidden layer followed by a softmax layer to obtain probabilities for classification. We used Keras for this model.

3.2.2 Long Short-Term Memory-I (LSTM-I)

We experiment with two Long Short-term Memory (Hochreiter and Schmidhuber, 1997) based approaches. In the first approach, we use an architecture similar to (Gupta et al., 2017) Here, similar to the CNN model, the input contains an entire dialogue. We experiment with two embedding layers, one with SSWE embeddings, and the other with GloVe embeddings. Figure 4 presents detailed description. Gupta et al. showed that SSWE embeddings capture sentiment information and GloVe embeddings capture semantic information in the continuous representation of words. Similar to the CNN model, here also, we input the word indices of dialogue. We pad input sequences with zeros so that each sequence has length n.

The architecture consists of two LSTM layers after each embedding layer. The LSTM layer outputs a vector of shape 128×1. Further, concatenation of these output vectors results a vector of shape 256×1. In the end, we have a hidden layer followed by a softmax layer. The output from the softmax layer is a vector of shape 4×1 which refers to class probabilities for the four classes. We used Keras for this model.

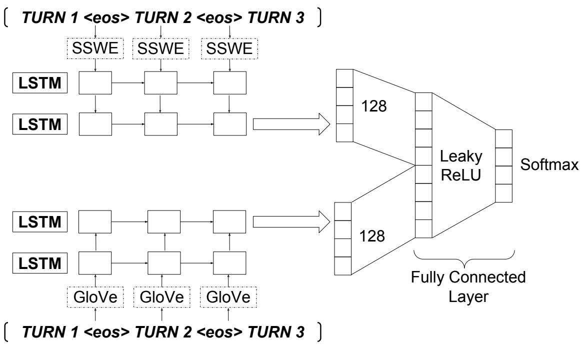


Figure 4: LSTM-I architecture.

3.2.3 Long Short-Term Memory-II (LSTM-II)

In the second approach, we use the architecture shown in Figure 5. This model consists of embedding layers, LSTM layers, a dense layer and a softmax layer. Here, the entire dialogue is not passed at once. Turn 1 is passed through an embedding layer which is followed by an LSTM layer. The output is a vector of shape 256×1. Turn 2 is also passed through an embedding layer which is followed by an LSTM layer. The output from Turn 2 is concatenated with the output of Turn 1 to form a vector of shape 512×1. The concatenated vector is passed through a dense layer which reduces the vector to 256×1. Turn 3 is passed through an embedding layer which is followed by an LSTM layer. The output from Turn 3 is concatenated with the reduced output of Turn 1 & 2, and the resultant vector has shape 512×1. The resultant vector is passed through a dense layer and then a softmax layer to find the probability distribution across different classes. We used Pytorch for this model.

The motivation of this architecture was derived from the Task's focus to identify the emotion of Turn 3. Hence, this architecture gives more weight to Turn 3 while making a prediction and conditions the result on Turn 1 & 2 by concatenating their output vectors. The concatenated vector of Turn 1 & 2 accounts for the context of the conversation.

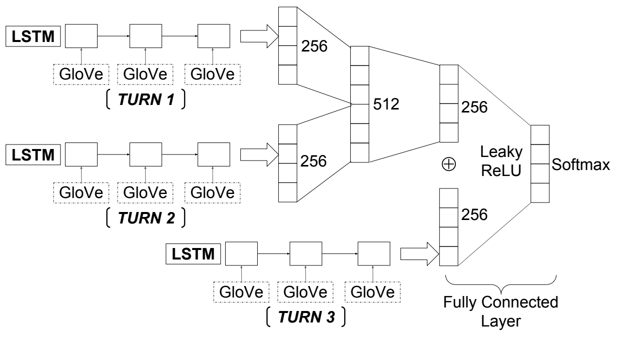


Figure 5: LSTM-II architecture.

4 Results

In Table 2, we report the performance of all the models described in the previous section. We train each model multiple (=5) times and compute the mean of scores.

Algorithm	$Score_{dev}$	$Score_{test}$
SVM	0.46	0.41
LR	0.44	0.40
SVM (tf-idf weighted averaging)	0.42	0.38
LR (tf-idf weighted averaging)	0.37	0.34
CNN	0.632	0.612
LSTM-I	0.677	**0.667**
LSTM-II	**0.684**	0.661

Table 2: Model performance on dev & test dataset.

SVM and Logistic Regression models did not yield very good results. We attribute this to the dialogue features that we use for the models. Tf-idf weighted average of GloVe vectors performed worse than the simple average of vectors. Hand-crafted features might have performed better than our current implementation. Neural network based models had very good results, CNN performed better than classical ML models but lagged behind LSTM based models. On the test set, our LSTM-I model performed slightly better than LSTM-II model.

Hyper-parameter selection for CNN was difficult, and we restricted to LSTM for the Phase 2 (i.e. test phase). We also noticed that the LSTM model was overfitting early in the training process (4-5 epochs) and that was a challenge when searching for optimal hyper-parameters. We used grid search to find the right set of hyper-parameters for our models. We grid searched over dropout (Srivastava et al., 2014), number of LSTM layers, learning rate and number of epochs. In case of the CNN model, number of filters was an extra hyper-parameter. We used Nvidia GeForce GTX 1080 for training our models.

5 Conclusion

In this paper, we experimented with multiple machine learning models. We see that LSTM and CNN models perform far better than classical ML methods. In phase-1 of the competition (dev dataset), we were able to achieve a score of 0.71, when the scoreboard leader had 0.77. But in phase-2 (test dataset), our best score was only 0.634, when the scoreboard leader had 0.79. After phase-2 ended, we experimented more with hyper-parameters and achieved an increase in scores on the test-set (mentioned in Table 2).

Full code for the paper can be found on GitHub*.

6 Future Work

Our scores on the test dataset suggest room for improvement. Now we are narrowing down to transfer learning where the starting point for our model will be a pre-trained network on a similar task. Our assumption is, this will help in better convergence on EmoContext dataset given the dataset size is not too large.

*https://github.com/lingo-iitgn/emocontext-19

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Stavros Giorgis, Apostolos Rousas, John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2016. aueb. twitter. sentiment at semeval-2016 task 4: A weighted ensemble of svms for twitter sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 96–99.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations. *CoRR*, abs/1707.06996.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 task 4: Sentiment analysis in twitter. In *Proceedings of the 10th international workshop on semantic evaluation (semeval-2016)*, pages 1–18.

BARBARA ORTUTAY. 2018. Poll: Teens prefer texting over face-to-face communication.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*.

Daily Telegraph Reporter. 2012. Texting more popular than face-to-face conversation.

Sebastian Ruder, Parsa Ghaffari, and John G Breslin. 2016. Insight-1 at semeval-2016 task 4: convolutional neural networks for sentiment classification and quantification. *arXiv preprint arXiv:1609.02746*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. volume 1, pages 1555–1565.

Xinyu Wang, Chunhong Zhang, Yang Ji, Li Sun, Leijia Wu, and Zhana Bao. 2013. A depression detection model based on sentiment analysis in micro-blog social network. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 201–213. Springer.

MARK WILSON. 2016. This startup is teaching chatbots real empathy.

Vikrant Yadav. 2016. thecerealkiller at semeval-2016 task 4: Deep learning based system for classifying sentiment of tweets on two point scale. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 100–102.

KGPChamps at SemEval-2019 Task 3: A deep learning approach to detect emotions in the dialog utterances.

[1]Jasabanta Patro, [2]Nitin Choudhary, [3]Kalpit Chittora, and [4]Animesh Mukherjee
IIT Kharagpur, India
{[1]jasabantapatro, [2]nitch13jan, [3]chittora.kalpit }@iitkgp.ac.in, [4]animeshm@cse.iitkgp.ac.in

Abstract

This paper describes our approach to solve *Semeval task 3: EmoContext*; where, given a textual dialogue, i.e., a user utterance along with two turns of context, we have to classify the emotion associated with the utterance as one of the following emotion classes: *Happy, Sad, Angry* or *Others*. To solve this problem, we experiment with different deep learning models ranging from simple LSTM to relatively more complex attention with Bi-LSTM model. We also experiment with word embeddings such as ConceptNet along with word embeddings generated from bi-directional LSTM taking input characters. We fine tune different parameters and hyper-parameters associated with each of our model and report the micro precision, micro recall and micro F1-score for each model. We identify the Bi-LSTM model, along with the input word embedding taken as the concatenation of the embeddings generated from the bidirectional character LSTM and ConceptNet embedding, as the best performing model with a highest micro-F1 score over the test set as 0.7261.

1 Introduction

In recent years, with the increase in the popularity of social media platforms, a significant amount of unstructured social media content (posts, tweets, messages etc.) has become available to the research community. People use social media as a platform to share their opinions, emotions, thoughts etc. This information has a huge potential to serve as a commercial catalyst to the business of companies and organizations, e.g., knowing the opinion of people about a product or a service could help the company to do betterment of their product or service according to the desire of the online consumers. In similar lines, emotions from the peoples' comments/opinion can help us to model the future popularity of the product or the service. Further, knowing public emotions about different events can help political parties to set their agenda for elections. Thus mining of opinions and emotions has a lot of practical relevance. Even prior to the social media era, emotion detection had achieved significant attention of psychologists and linguistics. An elaborate discussion of emotion as a research topic is presented in the next section.

In this paper, we describe our system and the models, with which, we achieved significant performance improvement over the SemEval baseline for task 3. The task is described in (Chatterjee et al., 2019), where, given a textual dialogue, i.e., a user utterance along with two turns of context, we have to classify the emotion associated with the utterance into one of the following emotion classes: *Happy, Sad, Angry* or *Others*. To solve this problem, we experiment with different deep learning models ranging from simple LSTMs to more complex attention based Bi-LSTM models. We also experiment with different word embeddings such as ConceptNet along with word embeddings generated from bi-directional character LSTMs. Our best model gives a micro F1 of 0.7261 on the test set released by the organizers.

2 Related works:

From the last decades of the previous century, emotion as a topic of research has captured the attention of many scientists and researches from different sub-fields of computer science and psychology. While prior to the current century, researches tried to capture emotions from acoustic signals (Murray and Arnott, 1993; Banse and Scherer, 1996) and facial expressions (Ekman and Friesen, 1971; Ekman, 1993; Ekman et al., 1987), in the current century, due to the emergence of

Internet and social media, expression and detection of emotion through/from texts and social media, has grabbed significant attention (Alm et al., 2005; Fragopanagos and Taylor, 2005; Binali et al., 2010; Dini and Bittar, 2016; Canales and Martínez-Barco, 2014; Seyeditabari et al., 2018) of researchers. The whole literature around emotion can be broadly divided into two categories (1) theoretical studies and (ii) computational studies.

Theoretical studies: The theoretical studies include searching answers for whether facial expressions of emotion are universal (Ekman and Friesen, 1971), searching for cross-cultural agreement in the judgment of facial expression (Ekman et al., 1987), studying the acoustic profile of vocal emotion expression (Banse and Scherer, 1996) etc. An exploratory discussion of the literature detailing human vocal emotion and its principal findings are presented in (Murray and Arnott, 1993). Application of the literature to the construction of a system capable of producing synthetic speech with emotion has also been discussed. A brief description of how emotion is processed in our brain is discussed in (LeDoux, 2000).

Computational studies: From last two decades detecting and analysis of emotion in texts and social media content has grabbed significant attention of computational linguists and social scientists. (Litman and Forbes-Riley, 2004) determine the utility of speech and lexical features for predicting student emotions in computer-human spoken tutoring dialogues. They develop an annotated corpora that mark each student dialogue for negative, neutral, positive and mixed emotions. Then they extract acoustic-prosodic features from the speech signal, and lexical items from the transcribed or recognized speech and apply machine learning approaches to detect the emotions. In the same year, (Busso et al., 2004) came up with an analysis of emotion recognition techniques, using facial expressions, speech and multimodal information etc. They conclude that the system based on facial expression gives better performance than the system based on just acoustic information for the emotions considered. Sentiment classification seeks to identify a piece of text according to its authors general feeling toward their subject, be it positive or negative. Traditional machine learning techniques have been applied to this problem with reasonable success, but they have been shown to work well only when there is a good match between the training and test data with respect to the topic. (Read, 2005) use emoticons to reduce dependency in machine learning techniques for sentiment classification. (Wiebe et al., 2005) came up with a corpus having an annotation of opinions, emotions, sentiments, speculations, evaluations and other private states in the language of 10000 lines. In the second half of the last decade several studies came up that analyze and detect (Fragopanagos and Taylor, 2005; Binali et al., 2010; Hancock et al., 2007; Strapparava and Mihalcea, 2008) emotion from the text using machine learning techniques of the text context. Detection of emotion over social media content (Yassine and Hajj, 2010; Pak and Paroubek, 2010; Gupta et al., 2010) and electronic media content (Neviarouskaya et al., 2007; Yang et al., 2007) started to become popular during this period. Emotion cause detection (Chen et al., 2010) introduce another interesting problem in this period. In the current decade many problems in this domain have been introduced like emotion detection in code-switching texts (Wang et al., 2015), metaphor detection with topic transition, emotion and cognition in context (Jang et al., 2016), sentence and clause level emotion annotation and detection (Tafreshi and Diab, 2018), detecting emotion in social media contents (Roberts et al., 2012; Liew, 2014), detecting emotion in multilingual contexts (Das, 2011) etc. to name a few. Several corpora have been introduced having an annotation of emotions and other associated things such as emotion over multi-genre corpus (Tafreshi and Diab, 2018), emotion corpus of multi-party conversations (Hsu et al., 2018), a fine-grained emotion corpus for sentiment analysis (Liew et al., 2016), a dataset of emotion annotated tweets to understand the interaction between affect categories (Mohammad and Kiritchenko, 2018) etc. to name few. Simultaneously, methodological novelty in emotion detection is also an important contribution by researchers in the recent times; works like emotion detection by GRUs (Abdul-Mageed and Ungar, 2017), representation mapping (Buechel and Hahn, 2018), hybrid neural networks (Li et al., 2016) etc. are a few such latest techniques. A detail description of different hidden challenges present in emotion detection over social media content is present in (Dini and Bittar, 2016). Few survey papers (Canales and Martínez-Barco, 2014; Seyeditabari et al., 2018) describing different emotion analysis and detection methods adopted in past years also came up during this period.

3 Dataset and preprocesing

3.1 Dataset

The dataset consists of three parts, (i) training data, (ii) development data (dev set), and (iii) test data. The training dataset consists of 30k conversations, where each conversation contains three turns of user utterances. The dev set and the test set contains 2754 and 5508 conversations respectively. These have been collected and annotated by the organisers. All of the conversations are classified into four classes, 'angry', 'sad', 'happy' and 'others'. Training data consists of about 5k samples each from 'angry', 'sad', 'happy' class, and 15k samples from 'others' class, whereas, both dev and test sets have a real-life distribution, which is about 4% each of 'angry', 'sad', 'happy' class and the rest is 'others' class.

3.2 Preprocessing

Before feeding the conversations to our model, we perform the following operations on the text:

- The three turns of the conversation are joined to form a single sentence; also if there are multiple instances of punctuation, then we keep only a single instance. The joined utterance contains the conversations in the same order as that is given in the data set.

- Each emoji is replaced by its respective English translation. Example: ':-)' is replaced by 'happy'.

- All the possible English contractions are replaced by their expanded forms. for example: 'don't' is converted 'do not'.

- We use Ekphrasis toolkit (Baziotis et al., 2017) to normalize the occurrence of the URL, e-mail, percent, money, phone, user, time, date, and number in the comments. For example, URLs are replaced by 'url', and all occurrences of @someone are replaced by 'user'.

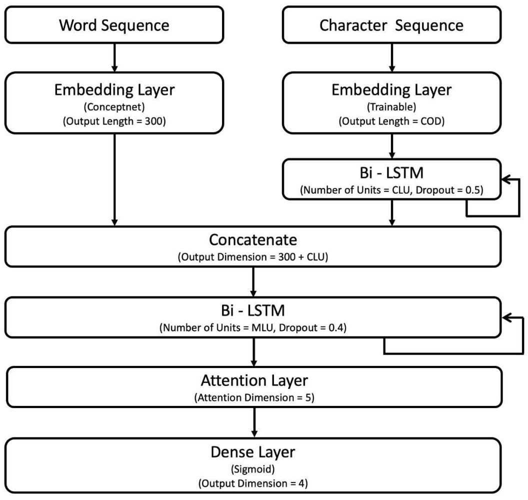


Figure 1: Overall schematic architecture of our system.

- Finally, we use NLTK Wordnet lemmatizer (Loper and Bird, 2002) to lemmatize the words to their roots.

4 System description

Our overall system is illustrated in Figure 1. We run different variants of our system by changing associated parameters, hyper-parameters and layers. For input, we consider a variety of options, which include (i) creating word embeddings using a Bi-LSTM trained on the character sequence of the sentence/utterance, (ii) using a pre-trained word embedding, i.e., Conceptnet, and (iii) concatenating (i) and (ii). From the architecture point of view our systems can be categorized into three types – (i) simple LSTM model, where an LSTM layer is considered instead of a Bi-LSTM layer (see figure 1) with no attention, i.e., the final hidden vector of LSTM layer is fed to the dense layer bypassing the attention layer (ii) simple Bi-LSTM model, where no attention layer is present, i.e., the final hidden vector of Bi-LSTM layer is fed to the dense layer by-passing the attention layer in figure 1, and (iii) Bi-LSTM model + attention, where we keep the attention layer active as shown in figure 1. We use the python module keras for our implementation.

5 Models and results

As previously stated, we experiment with different variants of the model. In this section, we discuss some of the top performing models and their performance. The results for different sys-

Model Type	Acc_μ	Pre_μ	Rec_μ	$F1_\mu$	$F1_{test}$
LSTM + Conceptnet	0.90	0.88	0.90	0.89	0.6825
Bi-LSTM + ConceptNet	0.90	0.86	0.91	0.88	0.6686
BiLSTM + (char embed. + Conceptnet)	0.90	0.88	0.89	0.89	**0.7261**
Bi-LSTM + (character embed. + Conceptnet) + no emojis	0.90	0.88	0.90	0.89	0.6418
Attentive Bi-LSTM + character embedding + Conceptnet	0.89	0.88	0.88	0.88	0.6900

Table 1: Results of different models; accuracy (Acc_μ), micro-precision (Pre_μ), micro-recall (Rec_μ) and micro-F1 score ($F1_\mu$) over the training data for five-fold cross validation; $F1_{test}$ is the micro-F1 score over the test set released by the organisers.

tems and their description are as shown in Table 1. We report two types of results (i) performance over training set which we obtain through five-fold cross-validation and (ii) performance over test data as reported by SemEval organizers. A short description of the model variants and their results are given below.

1. The first two models present in Table 1, as the name suggests, contains a layer of LSTM (1st model) and Bi-LSTM (2nd model). Sequence of words padded to a fixed length is given as input to this layer. The input sequence is then converted to an embedding vector with the help of pre-trained embedding matrices. We tried various pre-trained embedding matrices such as GloVe, fastText, ConceptNet and Google word2vec, out of which for Conceptnet we get best results. The outcome of LSTM/Bi-LSTM is given as input to the final dense layer which contains four nodes with sigmoid activation function for four emotions.

2. In the next model, we append character embeddings to the Conceptnet embeddings. This model produces the best performance over the test set, i.e., micro F1 score over the test set is 0.7261 as released by the organizers. The input to this model is a 2-D vector of words with characters in the second dimension.

3. The fourth model is the same as the previous model but emotion words (words which replaced emojis) are removed. As we can infer from the table this choice though did not affect the performance over the training set, the test set performance is significantly affected.

4. Finally, in the attentive Bi-LSTM model, we switch on the attention layer. Other parameters are kept the same as the third model model.

6 Conclusion

In this paper, we present a neural network based model to detect emotions from textual conversations. The usage of pre-trained embedding, Conceptnet gives a huge boost to the performance of our system. The performance reported in our paper could further be improved by implementing a better prepossessing pipeline and using more advanced RNN models. Furthermore, the dataset provided had a huge imbalance among different classes, therefore sampling among classes could result in increased performance. On the other hand, studying emotion in social media text can be linked further to the popularity of a product, service etc. which might be linked to financial interests of organizations. Further, how users expressing a particular predominant emotion interact with other users could be another line of future study.

References

Muhammad Abdul-Mageed and Lyle H. Ungar. 2017. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *ACL*.

Cecilia Ovesdotter Alm, Dan Roth, and Richard Sproat. 2005. Emotions from text: Machine learning for text-based emotion prediction. In *HLT/EMNLP*.

Rainer Banse and Klaus R. Scherer. 1996. Acoustic profiles in vocal emotion expression. *Journal of personality and social psychology*, 70 3:614–36.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754.

Haji Binali, Chen W Wu, and Vidyasagar Potdar. 2010. Computational approaches for emotion detection in text. *4th IEEE International Conference on Digital Ecosystems and Technologies*, pages 172–177.

Sven Buechel and Udo Hahn. 2018. Representation mapping: A novel approach to generate high-quality multi-lingual emotion lexicons. *CoRR*, abs/1807.00775.

Carlos Busso, Zhigang Deng, Serdar Yildirim, Murtaza Bulut, Chul Min Lee, Abe Kazemzadeh, Sungbok Lee, Ulrich Neumann, and Shrikanth Narayanan. 2004. Analysis of emotion recognition using facial expressions, speech and multimodal information. In *ICMI*.

Lea Canales and Patricio Martínez-Barco. 2014. Emotion detection from text : A survey.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Ying Chen, Sophia Yat Mei Lee, Shoushan Li, and Chu-Ren Huang. 2010. Emotion cause detection with linguistic constructions. In *COLING 2010*.

Dipankar Das. 2011. Analysis and tracking of emotions in english and bengali texts: a computational approach. In *WWW*.

Luca Dini and André Bittar. 2016. Emotion analysis on twitter: The hidden challenge. In *LREC*.

Paul Ekman. 1993. Facial expression and emotion. *The American psychologist*, 48 4:384–92.

Paul Ekman and Wallace V. Friesen. 1971. Constants across cultures in the face and emotion. *Journal of personality and social psychology*, 17 2:124–9.

Paul Ekman, Wallace V. Friesen, Maree O'Sullivan, Aryola Chan, I Diacoyanni-Tarlatzis, K G Heider, Rainer Krause, W A LeCompte, Tom K Pitcairn, and P. E. Ricci-Bitti. 1987. Universals and cultural differences in the judgments of facial expressions of emotion. *Journal of personality and social psychology*, 53 4:712–7.

Nickolaos F. Fragopanagos and John G. Taylor. 2005. Emotion recognition in human-computer interaction. *Neural networks : the official journal of the International Neural Network Society*, 18 4:389–405.

Narendra K. Gupta, Mazin Gilbert, and Giuseppe Di Fabbrizio. 2010. Emotion detection in email customer care. *Computational Intelligence*, 29:489–505.

Jeffrey T. Hancock, Christopher Landrigan, and Courtney Silver. 2007. Expressing emotion in text-based communication. In *CHI*.

Chao-Chun Hsu, Sheng-Yeh Chen, Chuan-Chun Kuo, Ting-Hao K. Huang, and Lun-Wei Ku. 2018. Emotionlines: An emotion corpus of multi-party conversations. *CoRR*, abs/1802.08379.

Hyeju Jang, Yohan Jo, Qinlan Shen, Michael Z. Miller, Seungwhan Moon, and Carolyn Penstein Rosé. 2016. Metaphor detection with topic transition, emotion and cognition in context. In *ACL*.

Joseph E. LeDoux. 2000. Emotion circuits in the brain. *Annual review of neuroscience*, 23:155–84.

Xiangsheng Li, Jianhui Pang, Biyun Mo, and Yanghui Rao. 2016. Hybrid neural networks for social emotion detection over short text. *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 537–544.

Jasy Suet Yan Liew. 2014. Expanding the range of automatic emotion detection in microblogging text. In *EACL*.

Jasy Suet Yan Liew, Howard R. Turtle, and Elizabeth D. Liddy. 2016. Emotweet-28: A fine-grained emotion corpus for sentiment analysis. In *LREC*.

Diane J. Litman and Katherine Forbes-Riley. 2004. Predicting student emotions in computer-human tutoring dialogues. In *ACL*.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit in proceedings of the acl workshop on effective tools and methodologies for teaching natural language processing and computational linguistics. *Philadelphia, Association for Computational Linguistics*, pages 62–69.

Saif Mohammad and Svetlana Kiritchenko. 2018. Understanding emotions: A dataset of tweets to study interactions between affect categories. In *LREC*.

Iain R. Murray and John L. Arnott. 1993. Toward the simulation of emotion in synthetic speech: a review of the literature on human vocal emotion. *The Journal of the Acoustical Society of America*, 93 2:1097–108.

Alena Neviarouskaya, Helmut Prendinger, and Mitsuru Ishizuka. 2007. Narrowing the social gap among people involved in global dialog: Automatic emotion detection in blog posts. In *ICWSM*.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*.

Jonathon Read. 2005. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *ACL*.

Kirk Roberts, Michael A. Roach, Joseph Johnson, Josh Guthrie, and Sanda M. Harabagiu. 2012. Empatweet: Annotating and detecting emotions on twitter. In *LREC*.

Armin Seyeditabari, Narges Tabari, and Wlodek Zadrozny. 2018. Emotion detection in text: a review. *CoRR*, abs/1806.00674.

Carlo Strapparava and Rada Mihalcea. 2008. Learning to identify emotions in text. In *SAC*.

Shabnam Tafreshi and Mona T. Diab. 2018. Sentence and clause level emotion annotation, detection, and classification in a multi-genre corpus. In *LREC*.

Zhongqing Wang, Sophia Yat Mei Lee, Shoushan Li, and Guodong Zhou. 2015. Emotion detection in code-switching texts via bilingual and sentimental information. In *ACL*.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39:165–210.

Changhua Yang, Kevin Hsin-Yih Lin, and Hsin-Hsi Chen. 2007. Emotion classification using web blog corpora. *IEEE/WIC/ACM International Conference on Web Intelligence (WI'07)*, pages 275–278.

Mohamed Yassine and Hazem M. Hajj. 2010. A framework for emotion mining from text in online social networks. *2010 IEEE International Conference on Data Mining Workshops*, pages 1136–1142.

KSU at SemEval-2019 Task 3: Hybrid Features for Emotion Recognition in Textual Conversation

Nourah Alswaidan **Mohamed El Bachir Menai**
Department of Computer Science
College of Computer and Information Sciences
King Saud University
Saudi Arabia
nourah_swaidan@yahoo.com, menai@ksu.edu.sa

Abstract

In this paper, we present the model submitted to the SemEval-2019 Task 3 competition: contextual emotion detection in text "EmoContext". We propose a model that hybridizes automatically extracted features and human engineered features to capture the representation of a textual conversation from different perspectives. The proposed model utilizes a fast gated-recurrent-unit backed by CuDNN (CuDNNGRU), and a convolutional neural network (CNN) to automatically extract features. The human engineered features take the term frequency-inverse document frequency (TF-IDF) of semantic meaning and mood tags extracted from SinticNet. For the classification, a dense neural network (DNN) is used with a sigmoid activation function. The model achieved a micro-F1 score of 0.6717 on the test dataset.

1 Introduction

Emotion recognition in text refers to the task of automatically assigning an emotion to a text selected from a set of predefined emotion labels. The SemEval-2019 competition (Chatterjee et al., 2019b) provides a textual dialogue and asks to classify the emotion as one of the emotion labels: happy, sad, and angry or others.

Previous research shows that emotion recognition has been performed on different types of text, including fairy tales[1] (Alm et al., 2005), news headlines[2] (Strapparava and Mihalcea, 2007), blog posts[3] (Aman and Szpakowicz, 2007), and tweets[4] (Mohammad et al., 2018). Whether a text expresses a single emotion or multiple emotions, it is challenging to recognize implicit emotions, which requires natural language understanding (NLU). Recognizing emotions in textual conversation increases difficulty by adding a dialogue format. Understanding emotions in textual conversation will further boost the research on NLU.

In this paper, we present an emotion recognition model that hybridizes human engineered features and automatically extracted features. For the human engineered features, we opted for calculating the term frequency-inverse document frequency (TF-IDF) of semantic meaning and mood tags retrieved from SenticNet. For the automatically extracted features, we explored two deep neural networks, a fast gated-recurrent-unit backed by CuDNN (CuDNNGRU) and convolutional neural networks (CNN). The classification is performed by a dense neural network (DNN).

The remainder of this paper is organized as follows. Section 2 describes the task corpus. Section 3 presents the proposed emotion recognition model. Section 4 presents the experimental results, and the main conclusions and future work are presented in Section 5.

2 Corpus

The organizers of the competition split the corpus into three datasets: a training dataset with 30160 instances, a development dataset with 2755 instances, and a test dataset with 5509 instances. The corpus was in a (.txt) format and contained five columns. The first column held the ID of the instances. The second, third and fourth columns held a conversation between two individuals. The first individual started the conversation then it was the second individuals turn, then the turn returned to the first individual. The fifth column held the emotion labels of the third turn in the conversation. The emotion label was either happy, sad, angry, or others. The distribution of the emotion

[1]http://people.rc.rit.edu/ coagla/affectdata/index.html
[2]http://web.eecs.umich.edu/ mihalcea/affectivetext
[3]http://saimacs.github.io
[4]https://competitions.codalab.org/competitions/17751

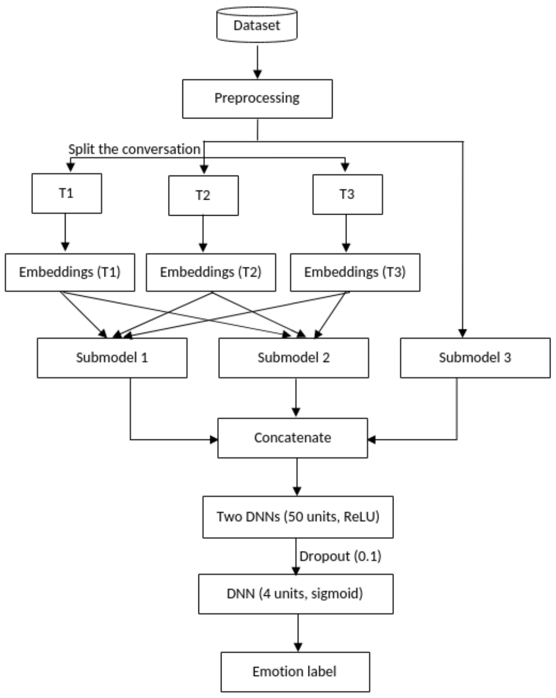


Figure 1: Diagram of the proposed model.

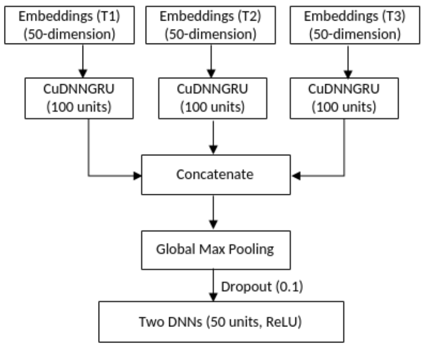

Embeddings (T1) (50-dimension)
Embeddings (T2) (50-dimension)
Embeddings (T3) (50-dimension)
CuDNNGRU (100 units)
CuDNNGRU (100 units)
CuDNNGRU (100 units)
Concatenate
Global Max Pooling
Dropout (0.1)
Two DNNs (50 units, ReLU)

Figure 2: Diagram of submodel 1.

labels differed between the training, development and test datasets. The training data consisted of approximately 5000 instances each of happy, sad, and angry labels, and 15000 instances of the others label. The development and the test datasets had 4% each of happy, sad, and angry labels and the rest was for the label others. During the competition, the development dataset and the test dataset were released without the label column. The full development dataset was released when the final evaluation on the test dataset started. The full test dataset was released after the end of the competition.

3 Proposed Model

In this section, we present the submitted emotion recognition model. Figure 1 shows an overview of the model.

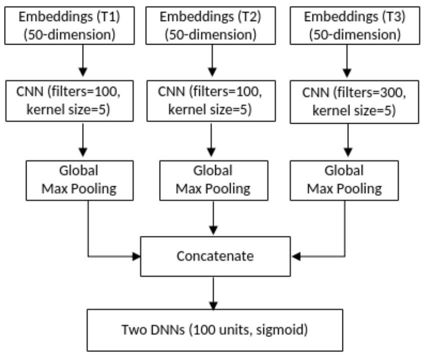

Embeddings (T1) (50-dimension)
Embeddings (T2) (50-dimension)
Embeddings (T3) (50-dimension)
CNN (filters=100, kernel size=5)
CNN (filters=100, kernel size=5)
CNN (filters=300, kernel size=5)
Global Max Pooling
Global Max Pooling
Global Max Pooling
Concatenate
Two DNNs (100 units, sigmoid)

Figure 3: Diagram of submodel 2.

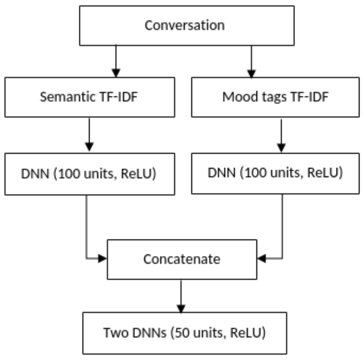


Figure 4: Diagram of submodel 3.

3.1 Preprocessing

The conversation style was informal and similar to a social media style of writing. Therefore, we utilized the ekphrasis[5] (Baziotis et al., 2017) tool. Ekphrasis was developed as part of the text processing pipeline for SemEval-2017 Task 4, sentiment analysis in Twitter. The preprocessing steps include Twitter-specific tokenization, unpack contractions, spell correction, word normalization, word annotation, word segmentation (for splitting hashtags), and replacing emoticons with suitable keywords.

We also grouped the most popular emojis into four classes, which matched the corpus emotion labels. With the use of regular expressions, we replaced the emoji with a keyword that represented the group the emoji belonged to. Then, we performed stopword removal and lemmatization with the use of the natural language toolkit[6] (NLTK).

[5]https://github.com/cbaziotis/ekphrasis
[6]https://www.nltk.org

3.2 Automatically Extracted Features

We utilized different deep neural networks, from the Keras[7] deep learning library, to enhance the representation of the text. However, we did not utilize any pretrained embeddings.

After text preprocessing, we split the text based on the conversation turns into turn 1 (T1), turn 2 (T2) and turn 3 (T3). An embedding matrix was generated for each turn of the conversation. Then, we applied BatchNormalization. These embeddings were used in two parallel submodels.

Submodel 1 in Figure 2, shows that each embedding matrix formed an input to a separate CuDNNGRU. The outputs of the three CuDNNGRUs were concatenated, and global max-pooling was performed. A dropout of value 0.1 was added to help avoid overfitting. Finally, the output was fed into two dense neural networks (DNN) with 50 units and a rectified linear unit (ReLU) activation function.

Submodel 2 in Figure 3, shows that each embedding matrix formed an input to a separate CNN with a sigmoid activation function. The number of filters of the first two CNNs was 100, but the third one had 300 filters, and the kernel size was five in all three CNNs. Next, global max-pooling was performed on the output of each CNN. Finally, the outputs were concatenated and fed into two DNNs with 100 units and a ReLU activation function.

3.3 Human Engineered Features

We took the conversation as a whole and extracted the following features:

- The TF-IDF of the Mood tags: SenticNet[8] (Cambria et al., 2018) was used to retrieve the mood tag of each word in the dataset. Then, every word was replaced by its mood tag. If a word had no mood tag, then it was deleted. Finally, the TF-IDF was calculated using the scikit-learn[9] library.

- The TF-IDF of the semantic meaning: SenticNet[8] was used to retrieve the semantic meaning of each word in the dataset. Then, the word was replaced by its semantic meaning. Finally, the TF-IDF was calculated using the scikit-learn[8] library.

[7]https://keras.io

[8]https://sentic.net

[9]https://scikit-learn.org

Item	Precision	Recall	F1
Angry	0.6345	0.8333	0.7205
Happy	0.5263	0.7746	0.6268
Sad	0.4641	0.7760	0.5808
Micro Average	0.5398	0.7962	0.6434

Table 1: Performance results on the development dataset using the automatically extracted features only.

Item	Precision	Recall	F1
Angry	0.4359	0.7933	0.5626
Happy	0.2734	0.5141	0.3570
Sad	0.4934	0.6000	0.5415
Micro Average	0.3858	0.6403	0.4815

Table 2: Performance results on the development dataset using the human engineered features only.

Item	Precision	Recall	F1
Angry	0.6531	0.8533	0.7399
Happy	0.5385	0.7887	0.6400
Sad	0.6216	0.7360	0.6740
Micro Average	0.6014	0.7962	0.6852

Table 3: Performance results on the development dataset using both automatically extracted features and human engineered features.

Item	Precision	Recall	F1
Angry	0.6456	0.7886	0.7100
Happy	0.5306	0.7324	0.6154
Sad	0.6780	0.7160	0.6965
Micro Average	0.6098	0.7476	0.6717

Table 4: Performance results on the test dataset using both automatically extracted features and human engineered features.

Submodel 3 in Figure 4, was responsible for training the human engineered features. Each of the TF-IDF features was trained with a DNN with 100 units and a ReLU activation function. Then, the outputs were concatenated and fed into two DNNs with 50 units and a ReLU activation function.

3.4 Emotion Classification

The three submodels were concatenated and fed into two DNNs with 50 units and a ReLU activation function. Then, a dropout of value 0.1 was used. Finally, a DNN with four units and a sigmoid activation function was added as an output layer for the classification of the emotions.

4 Experiments

The code was implemented in Python. We used the following libraries: NLTK[6], scikit-learn[9], and Keras[7] deep learning library run on a GPU, with the TensorFlow[10] backend.

We found the best hyper-parameters by evaluating on the development dataset. We trained with a batch size of 32, for two epochs with Adam optimization and 0.0005 as a learning rate. Tables 1 and 2 show the performance results obtained on the development dataset when only the automatically extracted features, and the human engineered features were used, respectively. They show that automatically extracted features clearly lead to the best microaverage performance results (Precision=0.5398, Recall=0.7962, F1=0.6434) in comparison to those obtained with the human engineered features only (Precision=0.3858, Recall=0.6403, F1=0.4815).

Table 3 presents the microaverage results obtained with the proposed model on the development dataset when both kinds of features were used altogether. The model achieved its best precision and F1 results (precision=0.6014, F1=0.6852) and the same recall obtained with only the automatically extracted features (Recall=0.7962). These performance results demonstrate the effectiveness of the proposed model. It scored above the baseline (Chatterjee et al., 2019a) on the test dataset. Table 4 presents the microaverage results obtained (Precision=0.6098, Recall=0.7476, F1=0.6717).

5 Conclusion

In this paper, we proposed a model to address emotion recognition in textual conversation based on using automatically extracted features and human engineered features. The usefulness of the model was demonstrated by the experimental results obtained in terms of precision, recall, and F1 measures. In the future, we plan to investigate the impact of other features on the performance of the model, including affect lexicons and pretrained embedding models.

References

Cecilia Ovesdotter Alm, Dan Roth, and Richard Sproat. 2005. Emotions from text: Machine learning for text-based emotion prediction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 579–586, Stroudsburg, PA, USA. Association for Computational Linguistics.

Saima Aman and Stan Szpakowicz. 2007. Identifying expressions of emotion in text. In *Proceedings of the 10th International Conference on Text, Speech and Dialogue*, TSD'07, pages 196–205, Berlin, Heidelberg. Springer-Verlag.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at SemEval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada. Association for Computational Linguistics.

Erik Cambria, Soujanya Poria, Devamanyu Hazarika, and Kenneth Kwok. 2018. Senticnet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings. In *AAAI*, pages 1795–1802.

Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. 2019a. Understanding emotions in text using deep learning and big data. *Computers in Human Behavior*, 93:309–317.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019b. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Saif M. Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 task 1: Affect in tweets. In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, USA.

Carlo Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 70–74, Stroudsburg, PA, USA. Association for Computational Linguistics.

[10]https://www.tensorflow.org

LIRMM-Advanse at SemEval-2019 Task 3: Attentive Conversation Modeling for Emotion Detection and Classification

Waleed Ragheb[1,2] , **Jérôme Azé** [1,2] , **Sandra Bringay**[1,3] and **Maximilien Servajean**[1,3]

[1]LIRMM UMR 5506, CNRS, University of Montpellier, Montpellier, France
[2]IUT de Béziers, University of Montpellier, Béziers, France
[3]AMIS, Paul Valery University - Montpellier 3 , Montpellier, France
{First.Last}@lirmm.fr

Abstract

This paper addresses the problem of modeling textual conversations and detecting emotions. Our proposed model makes use of 1) deep transfer learning rather than the classical shallow methods of word embedding; 2) self-attention mechanisms to focus on the most important parts of the texts and 3) turn-based conversational modeling for classifying the emotions. Our model was evaluated on the data provided by the SemEval-2019 shared task on contextual emotion detection in text. The model shows very competitive results.

1 Introduction

Emotional intelligence has played a significant role in many application in recent years (Krakovsky, 2018). It is one of the essential abilities to move from narrow to general human-like intelligence. Being able to recognize expressions of human emotion such as interest, distress, and pleasure in communication is vital for helping machines choose more helpful and less aggravating behavior. Human emotions are a mental state that can be sensed and hence recognized in many sources such as visual features in images or videos (Boubenna and Lee, 2018), as textual semantics and sentiments in texts (Calefato et al., 2017) or even patterns in EEG brain signals (Jenke et al., 2014). With the increasing number of messaging platforms and with the growing demand of customer chat bot applications, detecting the emotional state in conversations becomes highly important for more personalized and human-like conversations (Zhou et al., 2018).

This paper addresses the problem of modeling a conversation that comes with multiple turns for detecting and classifying emotions. The proposed model makes use of transfer learning through the universal language modeling that is composed of consecutive layers of Bi-directional Long Term Short Term Memory (Bi-LSTM) units. These layers are learned first in sequence-to-sequence fashion on a general text and then fine-tuned to a specific target task. The model also makes use of an attention mechanism in order to focus on the most important parts of each text turn. Finally, the proposed classifier models the changing of the emotional state of a specific user across turns.

The rest of the paper is organized as follows. In Section 2, the related work is introduced. Then, we present a quick overview of the task and the datasets in Section 3. Section 4 describes the proposed model architecture, some variants and hyperparameters settings. The experiments and results are presented in Section 5. Section 6 concludes the study.

2 Related Work

Transfer learning or domain adaptation has been widely used in machine learning especially in the era of deep neural networks (Goodfellow et al., 2016). In natural language processing (NLP), this is done through Language Modeling (LM). Through this step, the model aims to predict a word given some context. This is considered as a vital and important basics in most of NLP applications. Not only because it tries to understand the long-term dependencies and hierarchical structure of the text but also for its open and free resources. LM is considered as unsupervised learning process which needs only corpus of unlabeled text. The problem is that LMs get overfitted to small datasets and suffer catastrophic forgetting when fine-tuned with a classifier. Compared to Computer Vision (CV), NLP models are typically more shallow and thus require different fine-tuning methods. The developing of the Universal Language Model Fine-tuning

(ULMFiT) (Howard and Ruder, 2018) is considered like moving from shallow to deep pre-training word representation. This idea has been proved to achieve CV-like transfer learning for many NLP tasks. ULMFiT makes use of the state-of-the-art AWD-LSTM (Average stochastic gradient descent - Weighted Dropout) language model (Merity et al., 2018). Weight-dropped LSTM is a strategy that uses a DropConnect (Wan et al., 2013) mask on the hidden-to-hidden weight matrices, as a means to prevent overfitting.

On the other hand, one of the recent trend in deep learning models is the attention Mechanism (Young et al., 2018). Attention in neural networks are inspired from the visual attention mechanism found in humans. The main principle is being able to focus on a certain region of an image with "high resolution" while perceiving the surrounding image in "low resolution", and then adjusting the focal point over time. This is why the early applications for attention were in the field of image recognition and computer vision (Larochelle and Hinton, 2010). In NLP, most competitive neural sequence transduction models have an encoder-decoder structure (Vaswani et al., 2017). A limitation of these architectures is that it encodes the input sequence to a fixed length internal representation. This cause the results going worse performance for very long input sequences. Simply, attention tries to overcome this limitation by guiding the network to learn where to pay close attention in the input sequence. Neural Machine Translation (NMT) is one of the early birds that make use of attention mechanism (Bahdanau et al., 2014). It has recently been applied to other problems like sentiment analysis (Ma et al., 2018) and emotional classification (Majumder et al., 2018).

3 Data

The datasets are collections of labeled conversations (Chatterjee et al., 2019b). Each conversation is a three turn talk between two persons. The conversation labels correspond to the emotional state of the last turn. Conversations are manually classified into three emotional states for *happy*, *sad*, *angry* and one additional class for *others*. In general, released datasets are highly imbalanced and contains about 4% for each emotion in the validation (development) set and final test set. Table 1 shows the number of conversations examples and emotions provided in the official released datasets.

Dataset	Data size	Happy	Sad	Angry
Training	30160	5191	6357	6027
Validation (Dev)	2755	180	151	182
Testing	5509	369	308	324

Table 1: Used datasets.

4 Proposed Models

4.1 Model Architecture

In figure 1, we present our proposed model architecture. The model consists of two main steps: encoder and classifier. We used a linear decoder to learn the language model encoder as we will discuss later. This decoder is replaced by the classifier layers. The input conversations come in turns of three. After tokenization, we concatenate the conversation text but keep track of each turn boundaries. The overall conversation is inputted to the encoder. The encoder is a normal embedding layer followed by AWD-LSTM block. This uses three stacked different size Bi-LSTM units trained by ASGD (Average Stochastic Gradient Descent) and managed dropout between LSTM units to prevent overfitting. The conversation encoded output has the form of $C_{Enc} = [T^1_{Enc} \oplus T^2_{Enc} \oplus T^3_{Enc}]$ where T^i is the ith turn in the conversation and $\oplus$ denotes a concatenation operation and $T^i_{Enc} = \{T^i_1, T^i_2, \ldots, T^i_{N_i}\}$. The sequence length of turn i is denoted by N_i. The size of T^i_j is the final encoding of the j's sequence item of turn i.

For classification, the proposed model pays close attention to the first and last turns. The reasons behind this are that the problem is to classify the emotion of the last turn. Also, the effect of the middle turn appear implicitly on the encoding of the last turn as we used Bi-LSTM encoding on the concatenated conversation. In addition to these, tracking the difference between the first and the last turn of the same person may be beneficial in modeling the semantic and emotional changes. So, we apply self-attention mechanism followed by an average pooling to get turn-based representation of the conversation. The attention scores for the ith turn S^i is given by:

$$S^i = Softmax\{W_i.T^i_{Enc}\} \qquad (1)$$

Where W_i is the weight of the attention layer of the i^{th} turn and S^i has the form of $S^i = \{S^i_1, S^i_2, ..., S^i_{N_i}\}$. The output of the attention layer is the scoring of the encoded turn sequence $O^i =$

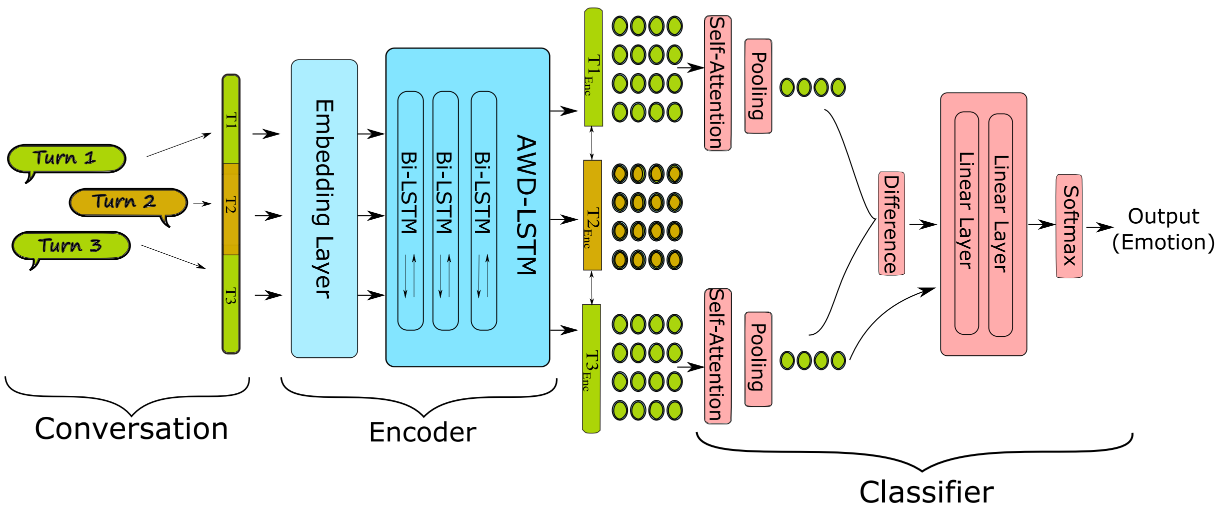


Figure 1: Proposed model architecture (*Model-A*).

$\{o_1^i, o_2^i, \ldots, o_{N_i}^i\}$ which has the same length as the turn sequence and is given by $O^i = S^i \odot T^i_{Enc}$ where $\odot$ is the element-wise multiplication. The difference of the pooled scored output of O^1 and O^3 is computed as O_{diff}. The Input of the linear block is X_{in} is formed by:

$$X_{in} = [O_{\text{diff}} \oplus O^3_{\text{pool}}] \quad (2)$$

The fully connected linear block consist of two different sized dense layers followed by a Softmax to determine the target emotion of the conversation.

4.2 Training Procedures

Training the overall models comes into three main steps: 1) The LM is randomly initialized and then trained by stacking a linear decoder in top of the encoder. The LM is trained on a general-domain corpus. This helps the model to get the general features of the language. 2) The same full LM after training is used as an initialization to be fine-tuned using the data of the target task (conversation text). In this step we limit the vocabulary of the LM to the frequent words (repeated more tan twice) of target task. 3) We keep the encoder and replace the decoder with the classifier and both are fine-tuned on the target task.

For training the language model, we used the Wikitext-103 dataset (Merity et al., 2016). We train the model on the forward and backward LMs for both the general-domain and task specific datasets. Both LMs -backward and forward- are used to build two versions of the same proposed architecture. The final decision is the ensemble of both. Our code is released at https://github.com/WaleedRagheb/Attentive-Emocontext.

4.3 Model Variations

In addition to the model - *Model-A* - described by Figure 1, we tried five different variants.

The first variant -(*Model-B*)- is formed by bypassing the self attention layer. This will pass the output of the encoder directly to the average pooling layer such that $X^B_{\text{in}} = [T_{\text{diff}} \oplus T^3_{\text{pool}}]$ where T_{diff} is the difference between the first and third pooled encoded turns of the conversations.

-(*Model-C*)- is to input a pooled condensed representation to the whole conversation C_{pool} rather than the last turn to the linear layer block. In this case: $X^C_{\text{in}} = [O_{\text{diff}} \oplus C_{\text{pool}}]$. We also studied two versions of the basic model where only one input is used $X^D_{\text{in}} = O_{\text{diff}}$ -(*Model-D*)- and $X^E_{\text{in}} = O^3_{\text{pool}}$ -(*Model-E*). In these two variants, we just change the size of the first linear layer.

Also, we apply the forward direction LM and classifier only without ensemble them with the backward direction and keep the same basic architecture -(*Model-F*).

4.4 Hyperparameters

We use the same set of hyperparameters across all model variants. For training and fine-tuning the LM, we use the same set of hyperparameter of AWD-LSTM proposed by (Merity et al., 2018) replacing the LSTM with Bi-LSTM. For classifier, we used masked self-attention layers and average pooling. For the linear block, we used hidden linear layer of size 100 and apply dropout of 0.4. We

Models	Results									
	Happy			Sad			Angry			Micro
	P	R	F1	P	R	F1	P	R	F1	F1
A	0.7256	0.7077	**0.7166**	0.8291	0.776	**0.8017**	0.7229	0.8054	**0.7619**	**0.7582**
B	**0.7341**	0.6514	0.6903	0.7401	**0.82**	0.778	0.7049	**0.8255**	0.7604	0.7439
C	0.7279	0.6972	0.7122	0.7765	0.792	0.7842	0.6941	0.8221	0.7527	0.7488
D	0.7214	**0.7113**	0.7163	0.8128	0.764	0.7876	0.6965	0.8087	0.7484	0.749
E	0.7204	0.7077	0.714	0.8205	0.768	0.7934	0.7026	0.8087	0.752	0.7512
F	0.7336	0.669	0.6998	**0.8377**	0.764	0.7992	**0.738**	0.7752	0.7561	0.75

Table 2: Test set results of the basic proposed model and it's variants.

used Adam optimizer (Dozat and Manning, 2017) with $\beta_1 = 0.8$ and $\beta_2 = 0.99$. The base learning rate is 0.01. We used the same batch size used in training LMs but we create each batch using weight random sampling. We used the same weights (0.4 for each emotion). We train the classifier on training set for 30 epochs and select the best model on validation set to get the final model.

5 Results & Discussions

The results of the test set for different variants of the model for each emotion is shown in table 2. The table shows the value of precision (P), recall (R) and F1 measure for each emotion and the micro-F1 for all three emotional classes. The micro-F1 scores are the official metrics used in this task. *Model-A* gives the best performance F1 for each emotion and the overall micro-F1 score. However some variants of this model give better recall or precision values for different emotions, *Model-A* compromise between these values to give the best F1 for each emotion. Removing the self-attention layer in the classifier -*Model-B*- degraded the results. Also, inputting a condensed representation of the all conversation rather than the last turn -*Model-C*- did not improve the results. Even modeling the turns difference only -*Model-D*- gives better results over *Model-C*. These proves empirically the importance of the last turn in the classification performance. This is clear for *Model-E* where the classifier is learned only by inputting the last turn of the conversation. Ensemble the forward and backward models was more useful than using the forward model only -*Model-F*.

Comparing the results for different emotions and different models, we notice the low performance in detecting happy emotion. This validate the same conclusion of Chatterjee et.al in (2019a). The model shows a significant improvement over the EmoContext organizer baseline (F1: 0.5868). Also, comparing to other participants in the same task with the same datasets, the proposed model gives competitive performance and ranked 11th out of more than 150 participants. The proposed model can be used to model multi-turn and multi-parties conversations. It can be used also to track the emotional changes in long conversations.

6 Conclusions

In this paper, we present a new model used for Semeval-2019 Task-3 (Chatterjee et al., 2019b). The proposed model makes use of deep transfer learning rather than the shallow models for language modeling. The model pays close attention to the first and the last turns written by the same person in 3-turn conversations. The classifier uses self-attention layers and the overall model does not use any special emotional lexicons or feature engineering steps. The results of the model and it's variants show a competitive results compared to the organizers baseline and other participants. Our best model gives micro-F1 score of 0.7582. The model can be applied to other emotional and sentiment classification problems and can be modified to accept external attention signals and emotional specific word embedding.

Acknowledgement

We would like to acknowledge La Région Occitanie and Communauté d'Agglomération Béziers Méditerranée which finance the thesis of Waleed Ragheb.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*, volume abs/1409.0473.

Hadjer Boubenna and Dohoon Lee. 2018. Image-based emotion recognition using evolutionary algorithms. *Biologically Inspired Cognitive Architectures*, 24:70 – 76.

Fabio Calefato, Filippo Lanubile, and Nicole Novielli. 2017. Emotxt: A toolkit for emotion recognition from text. In *2017 Seventh International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*, pages 79–80. IEEE.

Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. 2019a. Understanding emotions in text using deep learning and big data. *Computers in Human Behavior*, 93:309 – 317.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019b. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. volume abs/1611.01734.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339.

R. Jenke, A. Peer, and M. Buss. 2014. Feature extraction and selection for emotion recognition from eeg. *IEEE Transactions on Affective Computing*, 5(3):327–339.

Marina Krakovsky. 2018. Artificial (emotional) intelligence. *Commun. ACM*, 61(4):18–19.

Hugo Larochelle and Geoffrey E Hinton. 2010. Learning to combine foveal glimpses with a third-order boltzmann machine. In *Advances in Neural Information Processing Systems 23*, pages 1243–1251.

Yukun Ma, Haiyun Peng, and Erik Cambria. 2018. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive lstm. In *Association for the Advancement of Artificial Intelligence (AAAI 2018)*.

Navonil Majumder, Soujanya Poria, Devamanyu Hazarika, Rada Mihalcea, Alexander F. Gelbukh, and Erik Cambria. 2018. Dialoguernn: An attentive rnn for emotion detection in conversations. *CoRR*, Association for the Advancement of Artificial Intelligence (AAAI 2019).

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations (ICLR)*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR*, abs/1609.07843.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1058–1066, Atlanta, Georgia, USA. PMLR.

Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2018. Recent trends in deep learning based natural language processing [review article]. In *IEEE Computational Intelligence Magazine*, volume 13, pages 55–75.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*, pages 730–739.

MILAB at SemEval-2019 Task 3: Multi-View Turn-by-Turn Model for Context-Aware Sentiment Analysis

Yoonhyung Lee, Yanghoon Kim, and Kyomin Jung
Seoul National University, Seoul, Korea
{cpi1234, ad26kr, kjung}@snu.ac.kr

Abstract

This paper describes our system for SemEval-2019 Task 3: EmoContext, which aims to predict the emotion of the third utterance considering two preceding utterances in a dialogue. To address this challenge of predicting the emotion considering its context, we propose a Multi-View Turn-by-Turn (MVTT) model. Firstly, MVTT model generates vectors from each utterance using two encoders: word-level Bi-GRU encoder (WLE) and character-level CNN encoder (CLE). Then, MVTT grasps contextual information by combining the vectors and predict the emotion with the contextual information. We conduct experiments on the effect of vector encoding and vector combination. Our final MVTT model achieved 0.7634 microaveraged F1 score.

1 Introduction

Sentiment analysis is a task of identifying emotional information from text materials and has been studied by various research fields since it can be applied to many applications such as public survey and market analysis. However, most studies in sentiment analysis have only focused on a single sentence (Socher et al., 2011) or a single document (Pang and Lee, 2005). It is still hard to predict the emotion of a sentence with extra contextual information because the emotion can be understood differently depending on its context. SemEval-2019 Task 3: EmoContext (Chatterjee et al., 2019) provides a dataset of dialogues which consist of three utterances between two users (Figure 1). Participants are required to predict the emotion of the third utterance among ‘Happy’, ‘Sad’, ‘Angry’, and ‘Others’, considering its context of two preceding utterances.

In this paper, we propose a Multi-View Turn-by-Turn (MVTT) model which encodes each utterance separately and combines the encoded vectors to get the contextual information. MVTT model first generates vectors from each utterance with two encoders: word-level Bi-GRU encoder (WLE) and character-level CNN encoder (CLE). The two-encoder strategy makes MVTT model more robust to the noisy texts which have a lot of typos and abbreviations. Then, MVTT extracts contextual information by combining the vectors and makes a prediction. We compare the MVTT model with some variants focusing on utterance vector encoding and utterance vector combination methods with microaveraged F1 score (F1) which is the main evaluation metric.

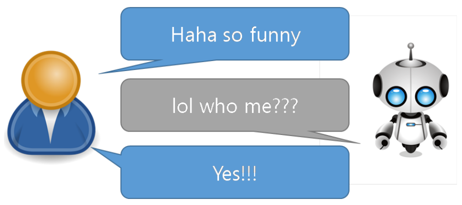


Figure 1: An example of a dialogue between two users.

This paper is organized as follows: Section 2 describes MVTT model architecture in detail. Section 3 describes dataset and various methods that we use to reflect the dataset’s characteristics to our training. Section 4 compares our results of MVTT model and other variants, and Section 5 outlines our conclusions.

2 System Description

This section describes our Multi-View Turn-by-Turn (MVTT) model which consists of two encoders: word-level Bi-GRU encoder (WLE) and character-level CNN encoder (CLE), which make our model more robust to noisy data. First, MVTT generates utterance vectors from each utterance using the encoders. Then, to understand the contextual information, MVTT combines the vectors into context-aware dialogue vectors and makes a

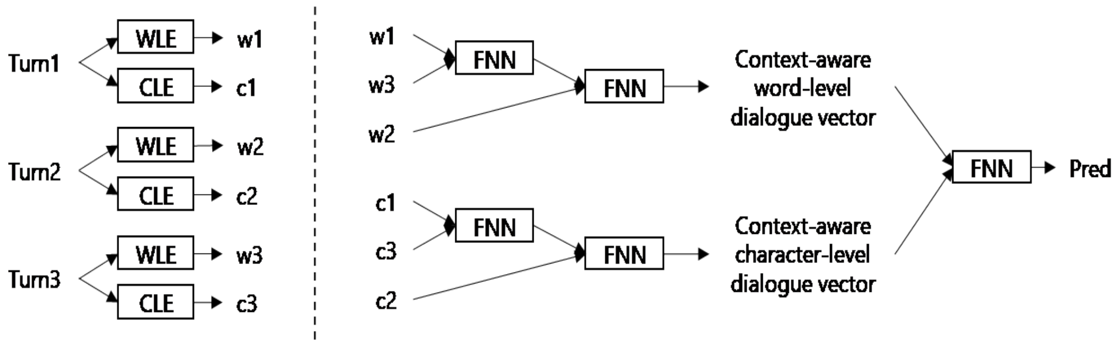

Turn1
WLE
w1
CLE
c1
Turn2
WLE
w2
CLE
c2
Turn3
WLE
w3
CLE
c3
w1
w3
w2
FNN
FNN
Context-aware word-level dialogue vector
c1
c3
c2
FNN
FNN
Context-aware character-level dialogue vector
FNN
Pred

Figure 2: Overview of MVTT model: how each encoder generates utterance vectors from each utterance and how MVTT model combines the vectors to make a prediction.

prediction using the context-aware dialogue vectors.

In this section, we first describe how each encoder generates the utterance vectors from each utterance and how MVTT combines the vectors into the context-aware dialogue vectors in detail.

2.1 Word-level Bi-GRU encoder

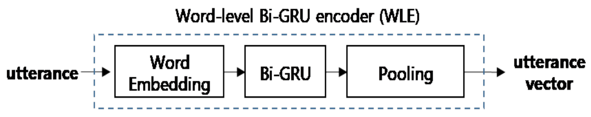


Figure 3: Word-level Bi-GRU encoder.

Word-level Bi-GRU encoder (WLE) takes an utterance as a sequence of word tokens $\{w_1, w_2, ..., w_N\}$. The tokens are fed into an embedding layer which is initialized with Glove word-embedding (Pennington et al., 2014) trained with a large twitter corpus. Then the embedded tokens are fed into the Bi-GRU encoder to get an utterance vector by max-pooling its hidden states over time.

WLE allows the model to benefit from pre-trained Glove word-embedding which have abundant information in syntactic and semantic word relationships. Also, MVTT benefits a lot from Bi-GRU encoder (Cho et al., 2014): (a) MVTT model can understand an text contextually with previous word information using Bi-GRU's gating mechanism; (b) by reading an text in two opposite ways, MVTT model can extract contextually more informative information. This is especially beneficial for MVTT to understand contextual meaning of a dialogue from each utterance.

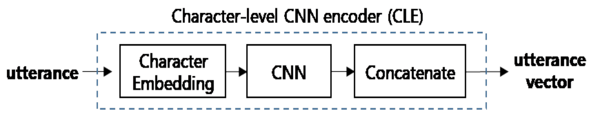


Figure 4: Character-level CNN encoder.

2.2 Character-level CNN encoder

Character-level CNN encoder (CLE) takes an input as a sequence of character tokens $\{c_1, c_2, ..., c_N\}$. The tokens are fed into a randomly initialized embedding layer, and then the embedded tokens are fed into the CNN encoder (Kim, 2014) to get an utterance vector.

Since it is impossible to consider all words in a word-level encoding, vocabulary consisting of 15k words is pre-defined based on word frequency and the other words are considered as out of vocabulary (OOV) tokens. Therefore, if the model only depends on WLE to extract features from a sequence of word tokens, a significant proportion of words will be tokenized as the OOV tokens when the texts are noisy. As a result, the model can't sufficiently utilize the benefit from pre-trained word-embedding. However, since the CNN encoder helps extract the local features from a sequence of character tokens, CLE enables our model to speculate the meaning of the text has some typos.

2.3 Multi-View Turn-by-Turn encoding

Multi-View Turn-by-Turn encoding is a method of generating utterance vectors from each utterance first and then combining the vectors into context-aware dialogue vectors. MVTT makes a prediction by encoding each utterance and combining them using concatenation and feed-forward neural network (FNN) as follows:

$$w13 = FNN([w1; w3])$$

$$w123 = FNN([w13; w2])$$

$$c13 = FNN([c1; c3])$$

$$c123 = FNN([c13; c2])$$

$$pred = FNN([w123; c123])$$

where $w1, w2, w3$ are the utterance vectors from each utterance generated by WLE and $c1, c2, c3$ are the utterance vectors from each utterance generated by CLE.

Since the first and third utterances are written by the same user, they are more informative in predicting the emotion of the third utterance. Therefore, by processing each utterance separately and combining them as above, the model can understand the context while maintaining the important emotional information.

2.4 Binary relevance classification

Binary relevance classification is a classification scheme to independently train binary classifiers for each label. It has usually been used for multi-label classification tasks and we apply the method to our task. In our system, we build three identical classifiers for 'Happy', 'Sad', 'Angry' classes and independently train them to output probabilities of each class. Then, we take the emotion with the highest probability as a class prediction, otherwise, if all the probabilities don't exceed 50%, take 'Others' as a predicted class.

3 Experiments

SemEval-2019 Task 3: EmoContext provided dialogue dataset consisting of three utterances written by two users and each sample is labeled among 'Happy', 'Sad', 'Angry' and 'Others'. In this section, we describe the dataset and some implementation details.

3.1 Datasets

The provided dialogue dataset is split into training, validation and test sets. Table 1 shows the label distribution of each data split. As Table 1 indicates, there are large differences in class label distributions among data splits and it is important to consider the differences in configuring our system.

Data split	Happy	Sad	Angry	Others
training	4243	5463	5506	14948
validation	142	125	150	2338
test	284	250	290	4677

Table 1: The statistics for the number of labels of each split.

3.2 Implementation details

We optimize our model using Adam optimizer (Kingma and Ba, 2014) and learning rate is set to 0.0015. We use Bi-GRU with 256 hidden units and CNN filters with window sizes of [3, 5, 9], 64 feature maps each. All FNN have 256 hidden units with $tanh$ activation function except for the last FNN classifier with $sigmoid$ function.

3.3 Pre-processing

In this task, we pre-process the utterances as described in Figure 5. We first lowercase all texts and replace abbreviations with their original forms as many as possible to make the best use of Glove word-embedding. Next, we unify emojis that have similar meanings into one specific emoji to help our model to learn emoji embeddings.

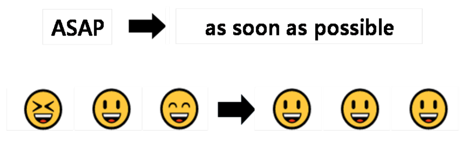


Figure 5: Text pre-processing.

3.4 Label smoothing

Label smoothing is a method to relax our confidence on the labels by using lower target values like 0.7 instead of 1. In the test set, almost all samples belong to the "Others" class with only a small percentage of examples belonging to the "Happy", "Sad", or "Angry" classes. Therefore, if we train each classifier for each emotion with label smoothing, we can prevent the model from predicting a emotion with excessive confidence and make the model be more likely to predict the emotions as 'Others'.

4 Results

In this section, we compare the performance of our MVTT model with some variants of our model. Since our model mainly consists of WLE and CLE, we try to investigate how our model benefits from both encoders. Further, we found that different combination methods of utterance vectors make a great difference in model evaluation. All of the results below are experimental results on the test set. MVTT outperforms all other variants and achieved 0.7634 microaveraged f1 score.

4.1 Ablation test on sentence embeddings

Our MVTT model utilizes the features of WLE and CLE. As is shown in Table 2, the model takes more advantage from WLE than CLE since the WLE utilizes the pre-trained word-embedding vectors trained on large twitter corpus which have abundant information in syntactic and semantic word relationships in the corpus. However, when we use both encoders, our MVTT model outperforms both models that use only one encoder. This results from the fact that CLE gives robustness to our model because it takes an input as a sequence of character tokens and extracts the local features from it.

Model	F1(H)	F1(S)	F1(A)	F1μ
WLE	0.7227	0.7680	0.7638	0.7515
CLE	0.6975	**0.7860**	0.7089	0.7270
MVTT	**0.7273**	0.7853	**0.7767**	**0.7634**

Table 2: Performance comparison among WLE, CLE and MVTT.

4.2 Impact of Turn-by-Turn encoding

Considering the characteristic of the given task, we find that the way to combine features from each utterance of a dialog is crucial. We tried several different combination methods, especially in the order of combination, to find out which setting has the most explainable structure with the best performance. We here list some variants with comparably better performance:

- C123: We simply concatenate $w1\,(c1)$, $w2\,(c2)$, $w3\,(c3)$ and feed it into a FNN to generate context-aware dialogue vectors.
- C12_3: Firstly concatenate $w1\,(c1)$, $w2\,(c2)$ and feed it into a FNN, and then concatenate the output and $w3\,(c3)$ and feed it into another FNN to generate context-aware dialogue vectors.
- C13_2 (MVTT): Firstly concatenate $w1\,(c1)$, $w3\,(c3)$ and feed it into a FNN, and then concatenate the output and $w2\,(c2)$ and feed it into another FNN to generate context-aware dialogue vectors.
- Submission: Ensemble of C123, C12_3, C13_2 models with various hyper parameters

Model	F1(H)	F1(S)	F1(A)	F1μ
C123	0.7119	0.7798	0.7602	0.7502
C12_3	0.7029	0.7674	0.7524	0.7406
Submission	0.7236	**0.7860**	0.7656	0.7581
MVTT	**0.7273**	0.7853	**0.7767**	**0.7634**

Table 3: The effect of vector combination on performance.

Table 3 shows the results of MVTT and some variants which combine the utterance vectors in other ways. As is shown in Table 3, our utterance vector combination method enables our model to understand both the emotional and contextual information. Since it is likely that a person's emotion is maintained through a 3-turn dialogue, combining the utterance vectors by user first and then making a prediction is beneficial to understand the context while maintaining emotional information.

5 Conclusion

In this paper, we propose a Multi-View Turn-by-Turn model (MVTT) for SemEval-2019 Task 3: EmoContext. Our goal was to predict the emotion of the third utterance in a dialogue consisting of three utterances. Firstly, MVTT model generates utterance vectors from each utterance using two encoders: word-level Bi-GRU encoder and character-level CNN encoder. The encoders make MVTT model more robust to the noisy texts. Then, MVTT combines the vectors to understand both the emotional and contextual meanings. We evaluated our MVTT model and its variants, focusing on utterance vector encoding and utterance vector combination. Our final MVTT model achieved 0.7634 microaveraged f1 score.

References

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger

Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing*, pages 151–161. Association for Computational Linguistics.

MoonGrad at SemEval-2019 Task 3: Ensemble BiRNNs for Contextual Emotion Detection in Dialogues

Chandrakant Bothe and Stefan Wermter
Knowledge Technology, Department of Informatics, University of Hamburg,
Vogt-Koelln-Str. 30, 22527 Hamburg, Germany
www.informatik.uni-hamburg.de/WTM/
{bothe, wermter}@informatik.uni-hamburg.de

Abstract

When reading "I don't want to talk to you any more", we might interpret this as either an angry or a sad emotion in the absence of context. Often, the utterances are shorter, and given a short utterance like "Me too!", it is difficult to interpret the emotion without context. The lack of prosodic or visual information makes it a challenging problem to detect such emotions only with text. However, using contextual information in the dialogue is gaining importance to provide a context-aware recognition of linguistic features such as emotion, dialogue act, sentiment etc. The SemEval 2019 Task 3 EmoContext competition provides a dataset of three-turn dialogues labeled with the three emotion classes, i.e. *Happy*, *Sad* and *Angry*, and in addition with *Others* as none of the aforementioned emotion classes. We develop an ensemble of the recurrent neural model with character- and word-level features as an input to solve this problem. The system performs quite well, achieving a microaveraged F1 score (F1µ) of 0.7212 for the three emotion classes.

1 Introduction

Humans might interpret text wrongly when reading sentences in the absence of context, so machines might too. When reading the following utterance,

```
Why don't you ever text me?
```

it is hard to interpret the emotion where it can be either a sad or an angry emotion (Chatterjee et al., 2019; Gupta et al., 2017). The problem becomes even harder when there are ambiguous utterances, for example, the following utterance:

```
Me too!
```

one cannot really interpret the emotion behind such an utterance in the absence of context. See Table 1 where the utterance "Me too!" is used in many emotional contexts such as *sad*, *angry*, and *happy* and also in the class *"others"* where none of aforementioned emotions is present.

Analyzing the emotion or sentiment of text provides the opinion cues expressed by the user. Such cues could assist computers to make better decisions to help users (Kang and Park, 2014) or to prevent potentially dangerous situations (O'Dea et al., 2015; Mohammad and Bravo-Marquez, 2017; Sailunaz et al., 2018). Character-level deep neural networks have recently showed outstanding results on text understanding tasks such as machine translation and text classification (Zhang et al., 2015; Kalchbrenner and Blunsom, 2013).

Usually, the utterances are short and contain mis-spelt words, emoticons, and hashtags, especially in the textual conversation. Hence, using character-level language representations can theoretically capture the notion of such texts. On the other hand, the EmoContext dataset is collected from the social media, and so the character language model used in our experiments is also trained on such a corpus (Radford et al., 2017).

We propose a system that encapsulates character- and word-level features and is modelled with recurrent and convolution neural networks (Lakomkin et al., 2017). We used our recently developed models for the context-based dialogue act recognition (Bothe et al., 2018). Our final model for EmoContext is an ensemble average of the intermediate neural layers, ended with a fully connected layer to classify the contextual emotions. The system performs quite well and we ranked on the public leaderboard (MoonGrad team) on CodaLab[1] in the top 35% of the systems (at the time of writing this paper Feb 2019) achieving the microaveraged F1 score (F1µ) of 0.7212 for the three emotion classes.

[1]https://competitions.codalab.org/competitions/19790

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 261–265
Minneapolis, Minnesota, USA, June 6–7, 2019. ©2019 Association for Computational Linguistics

	User 1	User 2	User 1	
id	**turn1**	**turn2**	**turn3**	**label**
2736	I don't hate you. you are just an AI	i don't hate anyone	me too	angry
2867	everything is bad	whats bad?	me too	sad
4756	I am very much happy :D	Thank you, I'm enjoying it :)	Me too	happy
8731	How r uh	am fine dear and u?	Me too	others

Table 1: Examples from training dataset, where *turn3* is mostly the same while contextual emotion is different.

Label	Train	Dev	Test
	30160	2755	5509
happy	4243	142	284
sad	5463	125	250
angry	5506	150	298
others	14948	2338	4677

Table 2: EmoContext Data Distribution; first row represents the total number of conversations in dataset.

2 Approach

The final model used for the submission to the EmoContext challenge is shown in Figure 1. It is an average ensemble of four variants of neural networks. *Net1* and *Net2* use the input from a pre-trained character language model; *Net3* and *Net4* use GloVe word embeddings as input. All models are trained with Adam optimizer at a learning rate of 0.001 (Kingma and Ba, 2014).

The dataset provided by the EmoContext organizers consists of the 3-turn dialogues from Twitter, where *turn1* is a tweet from user 1; *turn2* is a response from user 2 to that tweet, and *turn3* is a back response to user 2 (Gupta et al., 2017). The data distribution is presented in Table 2. We do not perform any special pre-processing except converting all the data into plain text.

2.1 Character-level RNN Model

The character-level utterance representations are encoded with the pre-trained recurrent neural network model[2] which contains a single multiplicative long short-term memory (mLSTM) (Krause et al., 2016) layer with 4,096 hidden units, trained on ∼80 million Amazon product reviews as a character-level language model (Radford et al., 2017). *Net1* and *Net2* are fed the last vector (LM) and the average vector (AV) of the mLSTM respectively. It is shown in (Lakomkin et al., 2017) that the AV contains effective features for emotion detection. The character-level RNN models (*Net1* and *Net2*) are identical and consist of two stacked bidirectional LSTMs (BiLSTM) followed by an average layer over the sequences computed by final BiLSTM.

2.2 Word-level RNN and RCNN Model

The word embeddings are used to encode the utterances. We use pre-trained GloVe embeddings (Pennington et al., 2014) trained on Twitter[3] with 200d embedding dimension. The average length of the utterances is 4.88 (i.e. ∼5 words/utterance on average) and about 99.37% utterances are under or equal to 20 words. Therefore, we set 20 words as a maximum length of the utterances. *Net3* is stacked with two levels of BiLSTM plus the average layer while *Net4* consists of a convolutional neural network (Conv). Conv in *Net4* over the embedding layer captures the meaningful features followed by a max pooling layer (max), with the kernel size of 5 with 64 filters and all the kernel weights matrix initialized with Glorot uniform initializer (Glorot et al., 2011; Kim, 2014; Kalchbrenner and Blunsom, 2013). The max pooling layer of pool size 4 is used in this setup, the output dimensions are shown in Figure 1. We build a recurrent-convolutional neural network (RCNN) model by cascading the stack of LSTMs and the average layer to model the context.

2.3 Ensemble Model

The overall model is developed in such a way that the outputs of all the networks (*Net1*, *Net2*, *Net3*, and *Net4*) are averaged and a fully connected layer (FCL) is used with *softmax* function over the four given classes. The complete model is trained end-to-end so that, given a set of three turns as an input, the model classifies the emotion labels.

[2]https://github.com/openai/generating-reviews-discovering-sentiment

[3]https://nlp.stanford.edu/projects/glove/

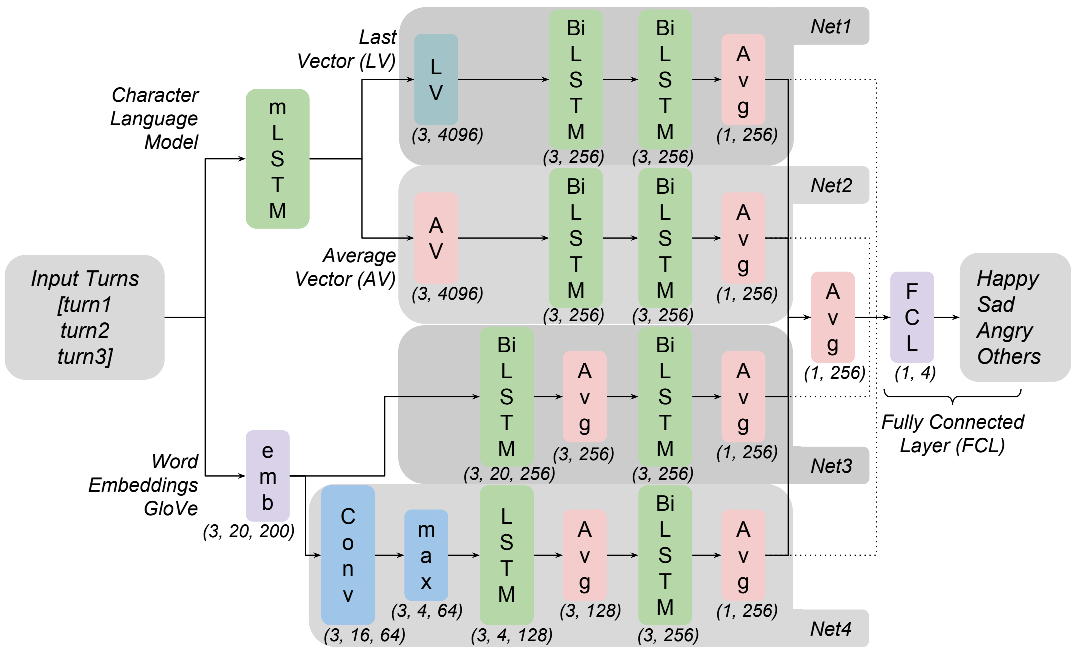


Figure 1: The overall architecture of the contextual emotion detection.

Models	F1µ
Baseline model (organizers)	0.5838
Our proposed model	**0.7212**
happy	0.6893
sad	0.7485
angry	0.7287

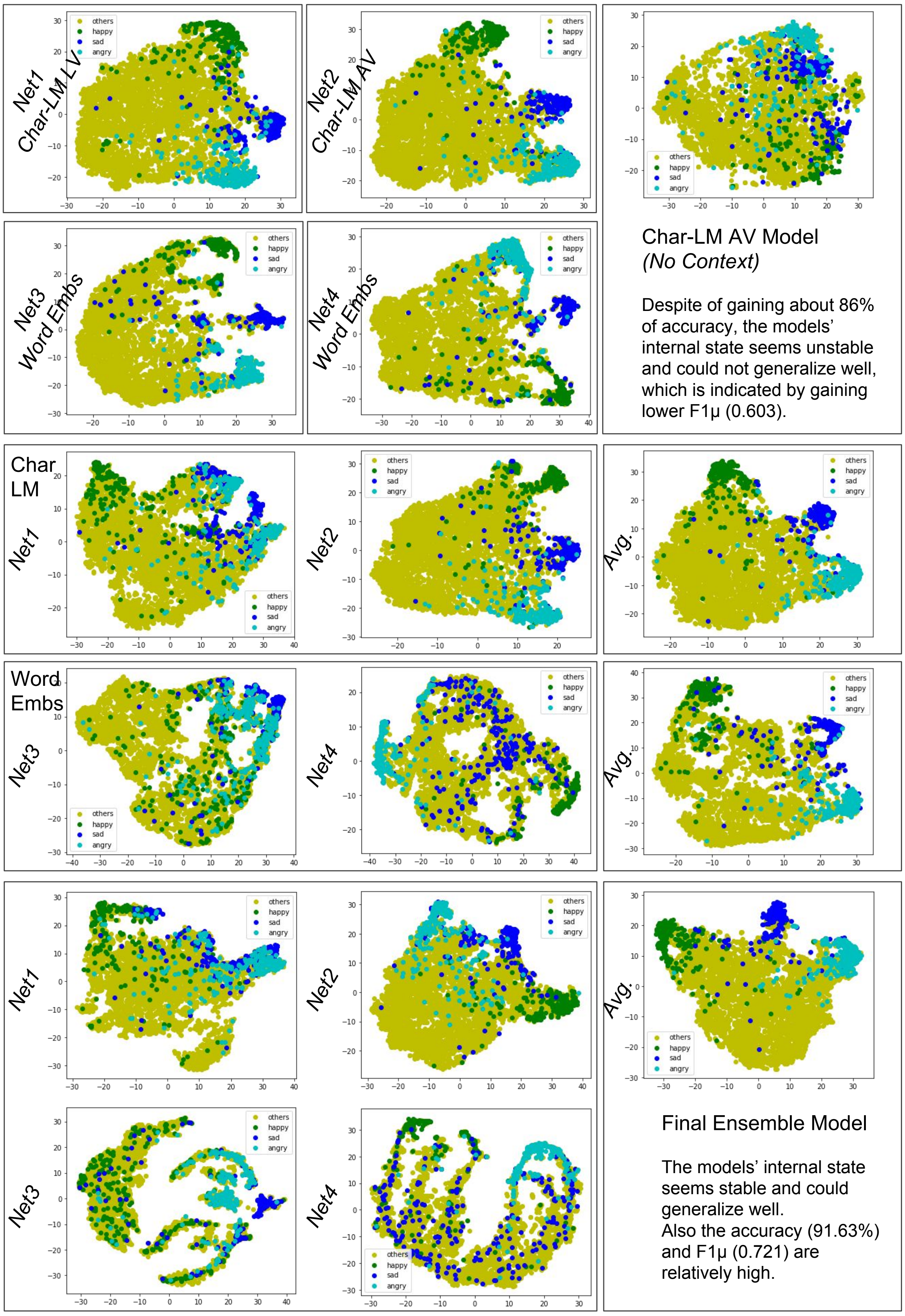
Table 3: Result as microaveraged F1 score (F1µ) compared to baseline and F1 score for each emotion.

3 Experiments and Results

The final submitted result to the challenge is shown in Table 3. The metric used for the challenge is the microaveraged F1 score (F1µ) for the three emotion classes, i.e. *Happy*, *Sad* and *Angry*. Our model performance was able compete quite well with the participating teams in the challenge. The main goal to present these experiments is to explore the features used for contextual emotion detection. For the comparison of different language features (character and word), we consider calculating the accuracy over all four classes, in addition to F1µ. The experimental setup developed and each network is tested individually and in an ensemble way. The results are reported in Table 4. When the models train individually, the output of the model being trained is directly connected to the FCL as shown in dotted line in Figure 1. From the results, it is clear that the average vector Char-LM AV Model outperforms the four individual networks. As this model performs well, we also train a single FCL to see the effect of the absence of context. The ensemble models, Char-LM Models (*Net1* and *Net2*) and Word Embs Models (*Net3* and *Net4*) show a clearer pick up on accuracy than individuals. The final ensemble model clearly improves the overall performance. However, we also ensemble the output predictions of all the networks trained individually, and average them at the end. Such ensembling is also effective for the overall improvement in the performance.

Models	Acc (%)	F1µ
Char-LM LV Model (*Net1*)	88.12	0.655
Char-LM AV Model (*Net2*)	89.87	0.694
Char-LM AV Model (*No Context*)	86.25	0.603
Word Embs Model (*Net3*)	88.27	0.665
Word Embs Model (*Net4*)	88.80	0.653
Char-LM Models (*Net1* and *Net2*)	89.59	0.688
Word Embs Models (*Net3* and *Net4*)	87.91	0.692
Final Ensemble Model	**91.63**	**0.721**
Avg. Ensemble Model *(outputs of individual nets)*	**91.71**	**0.721**

Table 4: Results comparing our experimental setups.



Char-LM AV Model
(No Context)

Despite of gaining about 86% of accuracy, the models' internal state seems unstable and could not generalize well, which is indicated by gaining lower F1µ (0.603).

Final Ensemble Model

The models' internal state seems stable and could generalize well.
Also the accuracy (91.63%) and F1µ (0.721) are relatively high.

Figure 2: Clustering the intermediate representations of different networks and their average (Avg.) ensembled representations. EmoContext test data is used to generate these representations.

In Figure 2, we demonstrate the intermediate representations taken at the last average layers of the networks on test data and plotted against four classes. We use t-SNE algorithm that converts multi-dimensional (in our case 256) to 2-dimensional arrays. We can notice that the *Net2* Char-LM AV model is quite consistent while other models are a bit unstable in clustering for the given emotions classes. For the final ensemble model, surprisingly, word models become too cluttered, but still contribute to the improvement.

4 Conclusion

The contextual emotion detection is a crucial step towards conversational analysis where emotion can aid the natural language understanding in socio-linguistic studies. Especially in the absence of facial expression and prosodic features, context becomes an important asset for emotion detection in the text. As we can see from the results our model could compete and provide insight to explore different feature representations. The ensemble modelling and transfer learning are effective tools for such a challenging task, specifically, when the given data is small and the labels are not balanced over all the samples.

Acknowledgments

This project has received funding from the European Unions Horizon 2020 research and innovation programme under the Marie Sklodowska Curie grant agreement No 642667 (SECURE).

References

Chandrakant Bothe, Sven Magg, Cornelius Weber, and Stefan Wermter. 2018. Conversational Analysis using Utterance-level Attention-based Bidirectional Recurrent Neural Networks. In *Proceedings of the International Conference INTERSPEECH 2018*.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. SemEval-2019 Task 3: EmoContext: Contextual Emotion Detection in Text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep Sparse Rectifier Neural Networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, PMLR*, volume 15, pages 315–323. PMLR.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A Sentiment-and-Semantics-Based Approach for Emotion Detection in Textual Conversations. *Proceedings of the Neu-IR 2017 SIGIR Workshop on Neural Information Retrieval*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Convolutional Neural Networks for Discourse Compositionality. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality, ACL*, pages 119–126.

Daekook Kang and Yongtae Park. 2014. Review-based measurement of customer satisfaction in mobile service: Sentiment analysis and VIKOR approach. *Expert Systems with Applications*, 41(4):1041–1050.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. *Proceedings of the Conference on EMNLP*, pages 1746–1751.

Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Ben Krause, Liang Lu, Iain Murray, and Steve Renals. 2016. Multiplicative LSTM for sequence modelling. *Workshop track of Proceedings of the International Conference on Learning Representations*.

Egor Lakomkin, Chandrakant Bothe, and Stefan Wermter. 2017. GradAscent at EmoInt-2017: Character and Word Level Recurrent Neural Network Models for Tweet Emotion Intensity Detection. In *Proceedings of the 8th Workshop WASSA at the Conference EMNLP*, pages 169–174. ACL.

Saif M. Mohammad and Felipe Bravo-Marquez. 2017. Emotion Intensities in Tweets. In *Proceedings of the Sixth Joint Conference on Lexical and Computational Semantics (*Sem)*, Vancouver, Canada.

Bridianne O'Dea, Stephen Wan, Philip J Batterham, Alison L Calear, Cecile Paris, and Helen Christensen. 2015. Detecting suicidality on Twitter. *Internet Interventions*, 2(2):183–188.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global Vectors for Word Representation. *Proceedings of the Conference on EMNLP*, pages 1532–1543.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to Generate Reviews and Discovering Sentiment. *arXiv: 1704.01444*.

Kashfia Sailunaz, Manmeet Dhaliwal, Jon Rokne, and Reda Alhajj. 2018. Emotion Detection from Text and Speech - A Survey. *Social Network Analysis and Mining*, 8(1):28.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *Advances in Neural Information Processing Systems*, pages 649–657.

NELEC at SemEval-2019 Task 3: Think Twice Before Going Deep

Parag Agrawal[*]
Microsoft
paragag@microsoft.com

Anshuman Suri[*]
Microsoft
ansuri@microsoft.com

Abstract

Existing Machine Learning techniques yield close to human performance on text-based classification tasks. However, the presence of multi-modal noise in chat data such as emoticons, slang, spelling mistakes, code-mixed data, etc. makes existing deep-learning solutions perform poorly. The inability of deep-learning systems to robustly capture these covariates puts a cap on their performance. We propose **NELEC** : **Ne**ural and **Le**xical **C**ombiner, a system which elegantly combines textual and deep-learning based methods for sentiment classification. We evaluate our system as part of the third task of 'Contextual Emotion Detection in Text' as part of SemEval-2019 (Chatterjee et al., 2019b). Our system performs significantly better than the baseline, as well as our deep-learning model benchmarks. It achieved a micro-averaged F_1 score of 0.7765, ranking 3^{rd} on the test-set leader-board. Our code is available at https://github.com/iamgroot42/nelec

1 Introduction

Sentiment analysis of textual data: Twitter data (Kouloumpis et al., 2011; Pak and Paroubek, 2010), movie reviews (Thet et al., 2010), and product reviews (Pang et al., 2008), is perhaps the most extensively explored problem, with a plethora of research to tackle it. Novel systems utilise deep learning architectures to achieve near-human performance on clean, well-formatted data. However, sentiment classification of chat data is significantly challenging. The presence of spelling errors, slang, emoticons, code-mixing, style of writing and abbreviations makes it significantly harder for existing deep-learning models to work on such data.

[*]Equal contribution, order determined by coin toss

Literature dealing with this problem comprises a wide range of approaches: from hand-crafted features to end-to-end deep-learning methods. Some rule-learning based methods use keyword-based analysis (Ko and Seo, 2000) and part-of-speech tagging (Agarwal et al., 2011). These procedures require extensive human-involvement for identifying keywords and designing rules and are thus not scalable.

Non-neural machine-learning methods utilize feature extraction algorithms like n-grams and Tf-Idf vectors, coupled with classification algorithms like Naive Bayes (Pang et al., 2002), Decision Trees (Bilal et al., 2016), SVM (Moraes et al., 2013). These approaches perform significantly better than rule-based approaches but fail to capture context well, since they ignore the order of words in text sequences.

Statistic	Train	Dev	Test
Emojis (%)	17.6	11.1	12.5
OOV (%)	3.7	4.9	4.9
OOV(processed) (%)	2.1	1.5	1.8
Avg.Length	13.6	12.7	12.7
Avg.Length(processed)	15.7	15.3	15.2
Happy emotion (%)	14.1	5.2	5.2
Sad emotion (%)	18.1	4.5	4.5
Angry emotion (%)	18.3	5.4	5.4

Table 1: Some statistics for the given training, development and test sets.

Neural, deep-learning based approaches use architectures such as variations of recurrent models: GRU (Chung et al., 2014), LSTM (Hochreiter and Schmidhuber, 1997), BiLSTM (Schuster and Paliwal, 1997) and Convolutional models (Mundra et al., 2017), performing significantly better than other machine-learning techniques. Their ability to generalise and capture context over long se-

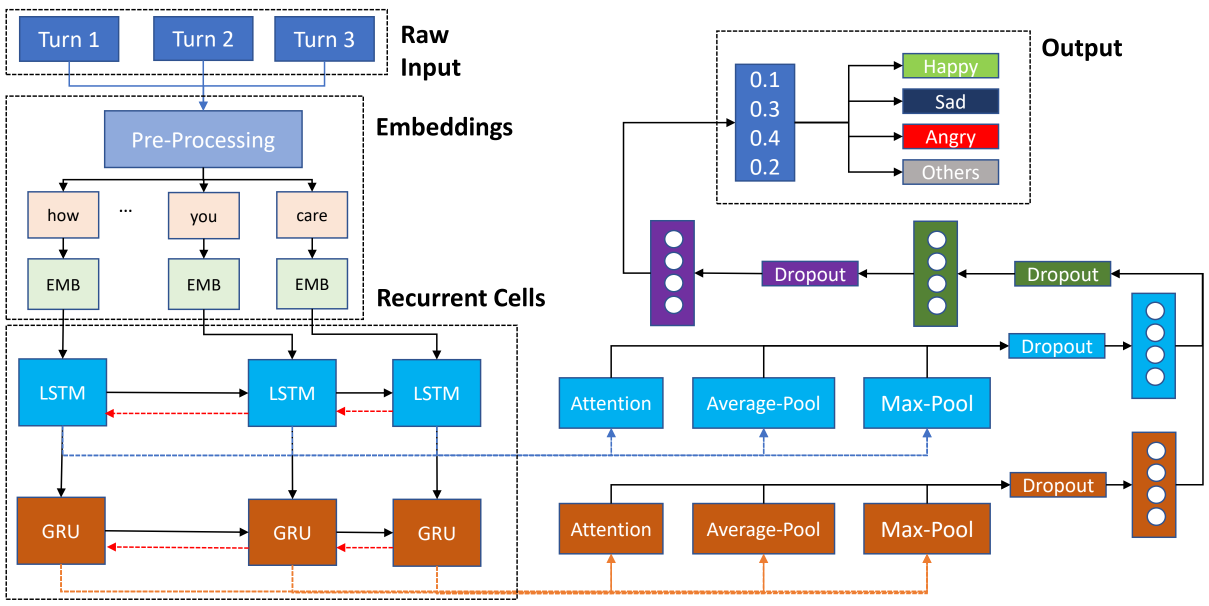


Figure 1: System diagram of the Deep-Learning model described in Section 2.1.

quences makes them a popular choice for text classification tasks.

We propose **NELEC**, a novel system specifically designed for sentiment classification. We combine lexical and neural features for sentiment classification, followed by class-specific thresholds for better labelling. Our system yields an F_1 score of 0.7765 on the test-set of Task 3 of SemEval 2019.

2 System Description

2.1 Deep Learning Model

We experiment with a two-layer, recurrent, deep-learning model with skip connections, bidirectional cells and attention (Figure 1). We trained our model for 100 epochs with Cyclic Learning Rate (Smith, 2017) scheduling. This model outperforms the baseline by a significant margin. An in-depth analysis of the cases where it fails reveals its shortcomings (along with that of a deep-learning model in general): it is not robust to misspellings and cannot capture the meaning of out-of-vocabulary words robustly. Even though pre-trained embeddings are available for most words, the context with which they are used in chat may vary from the corpora they were trained on, thus lowering their usability.

2.2 NELEC : Neural and Lexical Combiner

Since neural features have a lot of shortcomings, we shift our focus to lexical features. Using a combination of both lexical (n-gram features, etc.) and neural features (scores from neural classifiers), we trained a standard Light-GBM (Ke et al., 2017) Model for 100 iterations, with feature sub-sampling of 0.7 and data sub-sampling of 0.7 using bagging with a frequency of 1.0. We use $10^{-2} * \|weights\|_2$ as regularization. We also experimented with a logistic regression model, but it had a significant drop in performance for the 'happy' and 'angry' classes (Table 2). The total number of features used is 9270, out of which 9189 are sparse. The features we use in our model are described in the sections below:

2.2.1 Turn Wise Word *n*-Grams

Word level bi-grams and tri-grams (skip 1). These help capture patterns like “am happy” and automatically handles unseen data such as ”am very happy” or ”am so happy” because of the skip word. We take the term frequencies of these n-Grams as features. Word Grams not|good, hate, no|one had the highest feature gains.

2.2.2 Turn Wise Char *n*-Grams

Character level bi-grams and tri-grams. This feature helps capture character-level trends such as “haha” (and its variants), as well as emoticons. It helps with misspellings and makes the system robust to variants of several words like “haha”. h|a|h, w|o|w had one of the highest feature gains.

2.2.3 Valence Arousal Dominance

We used Valence-Arousal-Dominance data (Mohammad, 2018) in the following manner:

1. Mean of Valence and Arousal values, along with turn-wise Maximum Dominance value for all words. Turn 3 Arousal for maximum dominant word had the highest feature gain.

2. Turn-wise mean of Valence, Arousal and Dominance values.

2.2.4 Emotion Intensity

We use EmoLex (Mohammad and Turney, 2010), which associates words to eight emotions and two sentiments. For each turn, we obtain the number of words having specific emotions and sentiment and use it as a feature.

Model	F_1			
	happy	sad	angry	μ_{avg}
Without Data Pre-Processing				
Deep	.5863	.5977	.6485	.6123
NELEC	.7382	.8047	.7873	**.7765**
Logistic	.6712	.7642	.7151	.7154
Baseline	.5461	.6149	.5945	.5861
With Data Pre-Processing				
Deep	.5710	.6630	.7350	.6651
NELEC	.7324	.8015	.7878	**.7736**
Logistic	.6782	.7680	.7120	.7177
Baseline	.5797	.5973	.6241	.6024

Table 2: Class-wise and micro-averaged F_1 scores for NELEC, our deep-learning model and existing baseline.

2.2.5 Neural Features

We used scores obtained by utilizing available pre-trained classifiers features:

1. Scores obtained by running conversations through a Sentiment Classifier trained on Twitter Data using SSWE embeddings (Tang et al., 2014).

2. Signals from Adult and Offensive Classifiers (Yenala et al., 2017), obtained via the Text Moderation API by Microsoft Cognitive Services. As observed in Table 2, this helps in 'Anger' detection. [1]

[1]https://docs.microsoft.com/en-in/azure/cognitive-services/content-moderator/text-moderation-api

2.3 Lexical Count Features

Lastly, we used certain count features such as the number of interrogation marks, exclamation marks, uppercase letters, the total number of words and letters for each turn. These features were observed to be very helpful while detecting anger and happiness.

3 Data Preparation

The training, development and test sets consist of 30160, 2755 and 5509 examples respectively. The final model is trained on the combined training and development set. For each instance, one of four class labels: {happy, angry, sad, other}, is provided. Table 1 provides some statistics for the given dataset.

We concatenate all three turns per conversation. For the Deep-Learning approach, a special $\langle eos \rangle$ token is inserted in between these turn-conversations.

3.1 Pre-processing for NELEC

1. **Lemmatization**: Contrary to intuition, using lemmatization decreased the final performance of our model. Further analysis suggests that emotion is highly sensitive to exact words: information captured by the word "hate" and "hated" are very different, even though a lemmatization system would reduce them to the same word, and similarly for "happy" versus "happiest". Using lemmatization drops the system's F_1 score by 0.0092.

2. **WordNet for Synonyms**: We also tried using synonyms for nouns using the Wordnet Graph (Miller, 1998). However, a similar issue plagues this approach. For instance "dog", "doggie" and "puppy" are all synonyms, but they do not express the same kind of emotion: words like "puppy" convey much more positive emotion. Using Wordnet drops the system's F_1 score by 0.0023.

3. **Normalization**: We try word tokenization and normalization by removing diacritics, numbers, stop-words, question marks etc. However, this also drops the F_1 score by 0.0046.

Character n-gram features can handle lemmatization as well as misspellings for most of the cases without discarding any additional information. Finally, we only lower-cased the sentences.

Feature Dropped	Features (#)	$F1_{\mu avg}$	Angry F_1	Sad F_1	Happy F_1	$F1_{\mu avg}$ gain
Word n-grams	4565	.7355	.7373	.7723	.6995	.0410
Character n-grams	4624	.6067	.6271	.6168	.5749	**.1698**
Valence-Arousal	15	.7444	.7125	.7426	.7160	.0321
Word-emotion Classifier	30	.7537	.7584	.7739	.7301	.0228
Pre-Built Classifier	9	.7524	.7373	.7756	.7481	.0241
Lexical Count Features	27	.7654	.7751	.8015	.7217	.0111
Turn 1 (All Features)	2578	.7417	.7173	.7716	.7106	.0348
Turn 2 (All Features)	3873	.7642	.7719	.8015	.7217	.0123
Turn 1 & 2 (All Features)	6451	.7191	.7304	.7539	.6750	.0574

Table 3: Micro-averaged F_1 scores when all features apart from these (per row) are dropped. F_1 gain here refers to the gain when using the feature mentioned, as opposed to dropping it.

3.2 Pre-processing for Deep-Learning based Approach

We use pre-trained GloVe (Pennington et al., 2014) embeddings. Some observations are:

- **Emoticons**: Around 15% of all conversations includes at least one emoticon. We use embeddings from a pre-trained emoji2vec (Eisner et al., 2016) model to handle emoticons.
- **Words with repeated characters**: This trend is common for chat-data. For example, “heelloo”, “ookayy”. We design specific regular expressions to handle such variations.
- **Abbreviations and slang**: tokens such as “idk”, “irl” are converted to their full forms.

4 Experiments

To ascertain the novelty of our system, we report both class-wise and micro-averaged F_1 scores on the test set. We also compare our performance with the benchmarks provided by the contest organizers (Chatterjee et al., 2019a).

As mentioned in Section 3.2, data pre-processing on deep-learning models leads to significant performance gains, while leading to a drop in performance when using **NELEC**. **NELEC** outperforms both the baseline and our deep model by a considerable margin (Table 2).

4.1 Ablation Study

To analyze the usefulness of all features used by **NELEC**, we perform hold-one-out experiments on its features (Section 2.2). Results are reported in Table 3. There is a noticeable gain for most of the features, with character n-grams observing the maximum gain among them all.

One of the most intriguing patterns observed is the ease with which they detect sad emotion and an equal difficulty in detecting happiness.

- Words like “haha” and “okay” have several forms which all convey different magnitudes of emotion. While lemmatising such words, there is a significant loss of information.
- Most of the conversations labelled sad have easy-to-recognize signals such as negative emoticons, keywords like “lonely”, which make detection easy. On the other hand, differentiating *happy* and *others* is non-trivial.
- Not using the second turn, along with its associated features, leads to a negligible drop in F_1 performance. This observation highlights the importance of the first user (in data) in analyzing sentiment. Moreover, we can utilize this information to make the feature set even smaller, making the model smaller and faster.

5 Conclusion

We propose a deep neural architecture to solve the problem of emotion detection in conversations from chat data. Although it outperforms the existing baseline, its performance is not satisfactory. To better capture lexical features and make the model robust to misspellings, abbreviations, emoticons, etc., we propose **NELEC**, a **Ne**ural and **Le**xical **C**ombiner. Our model utilises lexical features, along with signals from pre-trained neural models for sentiment and adult-offensive classification to boost performance. Our system performs at par with the existing state of the art, yielding a micro-averaged F_1 score of 0.7765 on the test set, ranking 3^{rd} on the test-set leader-board.

References

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. 2011. Sentiment analysis of twitter data. In *Proceedings of the workshop on languages in social media*, pages 30–38. Association for Computational Linguistics.

Muhammad Bilal, Huma Israr, Muhammad Shahid, and Amin Khan. 2016. Sentiment classification of roman-urdu opinions using naïve bayesian, decision tree and knn classification techniques. *Journal of King Saud University-Computer and Information Sciences*, 28(3):330–344.

Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. 2019a. Understanding emotions in text using deep learning and big data. *Computers in Human Behavior*, 93:309–317.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019b. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota, USA.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. *arXiv preprint arXiv:1609.08359*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154.

Youngjoong Ko and Jungyun Seo. 2000. Automatic text categorization by unsupervised learning. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 453–459. Association for Computational Linguistics.

Efthymios Kouloumpis, Theresa Wilson, and Johanna D Moore. 2011. Twitter sentiment analysis: The good the bad and the omg! *Icwsm*, 11(538-541):164.

George Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Saif Mohammad. 2018. Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 english words. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 174–184.

Saif M. Mohammad and Peter D. Turney. 2010. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *HLT-NAACL 2010*.

Rodrigo Moraes, JoãO Francisco Valiati, and Wilson P GaviãO Neto. 2013. Document-level sentiment classification: An empirical comparison between svm and ann. *Expert Systems with Applications*, 40(2):621–633.

Shreshtha Mundra, Anirban Sen, Manjira Sinha, Sandya Mannarswamy, Sandipan Dandapat, and Shourya Roy. 2017. Fine-grained emotion detection in contact center chat utterances. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 337–349. Springer.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10, pages 1320–1326.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Leslie N Smith. 2017. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1555–1565.

Tun Thura Thet, Jin-Cheon Na, and Christopher SG Khoo. 2010. Aspect-based sentiment analysis of movie reviews on discussion boards. *Journal of information science*, 36(6):823–848.

Harish Yenala, Ashish Jhanwar, Manoj K Chinnakotla, and Jay Goyal. 2017. Deep learning for detecting inappropriate content in text. *International Journal of Data Science and Analytics*, pages 1–14.

NL-FIIT at SemEval-2019 Task 3: Emotion Detection From Conversational Triplets Using Hierarchical Encoders

Michal Farkas, Peter Lacko
Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovicova 2, 842 16 Bratislava, Slovakia
`michal.farkas@stuba.sk, peter.lacko@stuba.sk`

Abstract

In this paper, we present our system submission for the EmoContext, the third task of the SemEval 2019 workshop. Our solution is a hierarchical recurrent neural network with ELMo embeddings and regularization through dropout and Gaussian noise. We have mainly experimented with two main model architectures: simple and hierarchical LSTM network. We have also examined ensembling of the models and various variants of an ensemble. We have achieved microF1 score of 0.7481, which is significantly higher than baseline and currently the 19th best submission.

1 Introduction

Sentiment analysis has a long and successful history in the context of natural language processing. As with the majority of the problems in this domain, we have seen a gradual shift towards solutions based on neural models. Nowadays, such models can be readily used as a part of a larger solution, for example to analyse communication on social networks.

Then, perhaps unsurprisingly, the EmoContext task (Chatterjee et al., 2019b) with its format is highly evocative of the social networks. That is, it consists of conversational triplets, where the task is to correctly guess the emotion category of the last conversational turn.

Over the years, there was a number of different sentiment analysis and recognition competitions, workshops and shared tasks (Klinger et al., 2018; Rosenthal et al., 2017), however the conversational nature of the data is not common.

Our system is based on the recurrent neural networks, both simple and hierarchical architectures. We have experimented with various techniques and hyper-parameters, such as regularization, class weights, embeddings, strength of dropout and added noise. Finally, we have improved our results by creating an ensemble, with various voting methods and sample reweighing.

2 Approach

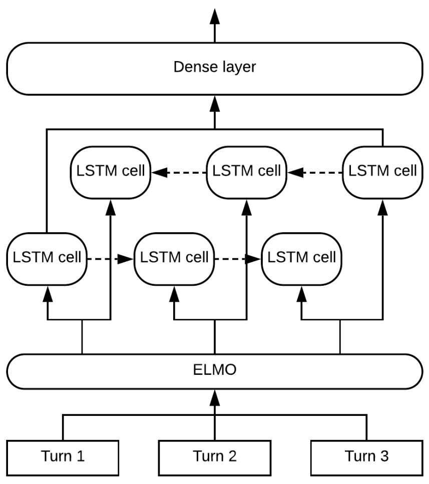
The first step in any natural language processing system is to preprocess the input data so it can be easily understood by the system. Since preprocessing was the major part of our previous work (Pecar et al., 2018), we have decided to prioritize work on the model instead. Nevertheless, we need to do some kind of preprocessing, hence we used the readily available Ekphrasis (Baziotis et al., 2017) tool.

We have experimented with both standard GloVe embeddings (Pennington et al., 2014) and more sophisticated ELMo embeddings (Peters et al., 2018). The improvements contextual embeddings, like ELMo, can bring are already well known and there is little need for additional comparisons, however the model that uses standard embeddings is considerably faster and hence more useful for quick experiments. Since training ELMo can be quite time consuming and resource intense, we have opted for pretrained models that are part of the AllenNLP library (Gardner et al., 2017).

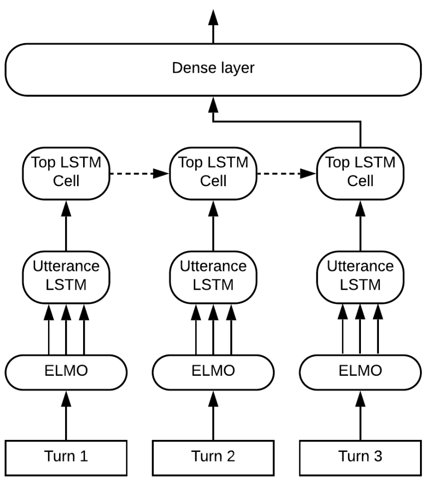
It should be noted that the character sensitive nature of the ELMo embeddings should help with typos, which were not fixed by preprocessing, and other similar errors.

2.1 Model

We have experimented with two main model variants. First model variant uses a simple bi-directional LSTM encoder (Hochreiter and Schmidhuber, 1997), second variant uses a hierarchical encoder, both can be seen in the figure 1. In both cases, the encoders are followed by dense



a) Simple encoder model.



b) Hierarchical encoder model.

Figure 1: Architecture of the main model variants.

layer which outputs probability distribution of labels.

Simple encoder works on concatenated utterances, utterances are part of a single string separated by semicolon. During the development we have also ran several runs only on the last utterance.

Hierachical model consists of two different encoders, at the utterance level and on the dialog level. This is fairly common practice in the dialog system domain (Serban et al., 2016). The utterance encoder is a bi-directional LSTM which encodes utterances into their representations. This utterance representation is then sent to the dialog encoder, which is a uni-directional LSTM.

Much of our model was dictated by a lack of GPU memory [1], we had to prioritize what to include in the model. Fully-connected layer as a dialog encoder, separate encoder for each turn, dense layer as a top encoder and attention mechanisms, did not achieve significantly better results and in some cases resulted in insufficient memory.

In the end, we had better results with the simpler, larger model rather than with the more complex, smaller model.

2.2 Training

We have opted to use Adam optimizer due to its far better performance on the hierarchical model where stochastic gradient descent did not perform up to our expectations.

[1]We used a single RTX 2070 with 8GB of memory

To properly regularize the model we have used the dropout (Srivastava et al., 2014) combined with the gaussian noise applied to word embeddings. Since we did not use multi-layer LSTMs we did not apply dropout in any other place in the model.

In the case of ensembles, we scheduled 90% reduction of the learning rate at the third epoch. This was done in the hopes of achieving less variance in the performance of various runs. Since a single model is trained considerably faster than an entire ensemble, we did not schedule learning rate change, as we were able to pick the best model or average from a variety of runs.

The class imbalance in the dataset, both for the train, validation and test set, was an issue we had to deal with. We have opted for class reweighing instead of a weighted sampling, due to ease of implementation. We have experimented with various weight setups.

2.3 Ensemble

To improve our results and help with significant variance in performance of different experiment runs, we have used an ensemble of multiple models. It should be noted that they are the exact same model, trained several times.

We have experimented both with voting through summation of the probabilities and with stacked classifier on top of the ensemble results. Stacked classifier is a single dense layer with the rectified linear activation and softmax. As input it takes a tensor of shape *(batch_size, ensem-*

Model	Size	Input data	Embeddings	MicroF1
Hierarchical	3076/1024	all	ELMo	**0.733**
Hierarchical	2048/1024	all	ELMo	**0.7213**
Hierarchical	1024/512	all	ELMo	**0.7172**
Simple	2048	concatenated	ELMo	**0.7123**
Simple	4096	concatenated	ELMo	**0.7079**
Simple	2048	last only	ELMo	**0.7076**
Hierarchical	2048/1024	all	GloVe	**0.6394**
Simple	4096	concatenated	GloVe	**0.639**
Simple	2048	concatenated	GloVe	**0.6312**
Hierarchical	1024/512	all	GloVe	**0.6294**
Hierarchical	3076/1024	all	GloVe	**0.6291**
Simple	2048	last only	GloVe	**0.5984**

Table 1: Comparison of various model variants.

Size	Voting	Reweighing	Max	Average	Combined
3	sum	No	0.7371	0.7265	**0.7481**(+0.0110/+0.0216)
5	sum	No	0.7426	0.7295	**0.7454**(+0.0028/+0.0159)
5	sum	Yes	0.7369	0.7272	**0.7430**(+0.0061/+0.0158)
3	sum	Yes	0.7306	0.7144	**0.7381**(+0.0075/+0.0237)
5	sum	Yes*	0.7281	0.7191	**0.7373**(+0.0093/+0.018)
3	stack	No	0.7248	0.7103	**0.7326**(+0.0078/+0.0223)
5	stack	Yes	0.7338	0.7270	**0.7310**(-0.0028/+0.004)
3	stack	Yes	0.7336	0.7257	**0.7289**(-0.0047/+0.0032)

Table 2: Comparison of various ensemble setups.

Distribution	Weights	MicroF1
train set	1.0/1.0/1.0/1.0	0.6924
test set	0.25/0.25/0.25/1.7	0.733
balanced_1	1.56/1.56/1.56/0.5	0.6888
balanced_2	1.56/1.56/1.56/0.3	0.6651
test w/o others	1.33/1.33/1.33/0.	0.2429

Table 3: Effect of different class weight setups.

*ble_size*num_labels)* and outputs a tensor of shape *(batch_size, num_labels)*.

To ensure that the single models in the ensemble will specialize on different samples, we have included the option for sample weight rebalance, based on their performance on the already trained models. However, error in the code caused that the rebalance calculation took into account only the last model and that the sample weights were gradually rising for the latter models in an ensemble. This was fixed only after the submissions were closed.

3 Evaluation

In this section, we cover metrics used, our experiments and analysis of our results. All results in this sections are achieved on the test set.

For evaluation we have modified code that is the part of the starter kit (Chatterjee et al., 2019a). Out of all metrics this function calculates we have primarily used microF1 score, which is the score reported in all our tables.

3.1 Results

In our experiments, we have explored a variety of different models, setups, ensembling approaches and effects of class weights. If not specified otherwise, models are using categorical crossentropy and following parameters:

- batch size: 32
- gaussian noise after embedding layer: 0.5 for simple, 3 for the hierarchical model
- dropout after embedding layer: 0.5 for simple, 0.6 for the hierarchical model

Class	Precision	Recall	F1
Angry	0.6948	0.8020	0.7445
Happy	0.7303	0.6866	0.7078
Sad	0.7687	0.8240	0.7954
Micro Average	0.7281	0.7692	0.7481

Table 4: Summary of the best submission.

Our first set of experiments are targeted at the model architecture and the effect of the used embeddings, results can be seen in table 1. In this table, all results are an average of three different runs. Unsurprisingly, the most significant effect is from the type of embeddings used. The effect of the rest of the factors seems to be in this order: model/input and size. At least for the ELMo embeddings, hierarchical models universally outperformed simple models. For the GloVe embeddings there is no clear separation, however the differences are rather small and if we averaged from more experiment runs a distinction could appear. The model that takes only the last turn into account was last when compared with the same embeddings, however we expected a more pronounced difference.

The next batch of experiments examines setup of our ensembles as seen in the table 2. In these experiments each row represent a single run of the entire ensemble. The combined score is the score of the ensemble after voting, in the parentheses we see change in respect to the max and average of the constituent models. Since the flawed reweighing does not seem to have a significant effect we have decided to left these experiments in. The experiment with asterisk, done after the submissions were closed, was run with the rebalancing fixed and while it shows second best improvement it is hard to tell if this is just a noise or a real effect. The most significant finding of this batch is that the stacked classifier performed rather poorly compared to the summation. For the stacked classifier, in two cases out of three, the combined score is actually worse than the best model in an ensemble.

Lastly, we have taken a look at the effect of different class weights, which can be seen in table 3, where the first column signifies distribution resulting from the given reweighing. We have experimented with ignoring others class due to the way the evaluation is done. The effect of such rebalancing is that all of the samples belonging to the ’others’ class is classified as one of the other classes, which causes extremely high recall. Since the test set distribution is closer to the distribution of the train set than to the balanced dataset, trying to reweigh the data to obtain a balanced dataset[2] is worse than doing nothing. The best results are obtained when the reweighed distribution is the same as the test set, even though score is not averaged over the ’others’ class.

Detailed summary of our best submission can be seen in the table 4.

4 Conclusion

In this paper, we have presented our models and experiments for emotion detection in conversational triples. We have also discussed results, various setups and model variants.

The bulk of our work was focused on simple vs. hierarchical models. We observed that the hierarchical model outperformed simple model. Additionally, simple model with only the last turn of conversation was only slightly worse than the model with the entire context.

We managed to improve our results by using an ensemble of multiple instances of the same model. During our experiments summation method of result combination proved to be superior to using stacked classifier. Interestingly, the stacked classifier could be perhaps a way to adapt model to different class distribution.

Our efforts were hindered by insufficient amount of memory on our GPU, thus we could not include every feature we wanted into our model. Perhaps, using GPU with more memory available, we could achieve slightly better results.

The source code of our system is available at GitHub [3].

Acknowledgements

This work was partially supported by the project Development of research infrastructure STU, project no. 003STU-2-3/2016 by the Ministry of Education, Science, Research and Sport of the Slovak Republic, Slovak Research and Development Agency under the contract No. APVV-16-0213 and APVV-16-0213 and by financial contribution from the STU Grant scheme for Support of Young Researchers.

[2]balanced_2 slightly supresses the others class

[3]`https://github.com/michalfarkas/nl-fiit_emocontext`

References

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada. Association for Computational Linguistics.

Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. 2019a. Understanding emotions in text using deep learning and big data. *Computers in Human Behavior*, 93:309–317.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019b. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Roman Klinger, Orphee De Clercq, Saif Mohammad, and Alexandra Balahur. 2018. Iest: Wassa-2018 implicit emotions shared task. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 31–42. Association for Computational Linguistics.

Samuel Pecar, Michal Farkaš, Marian Simko, Peter Lacko, and Maria Bielikova. 2018. Nl-fiit at iest-2018: Emotion recognition utilizing neural networks and multi-level preprocessing. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 217–223. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518, Vancouver, Canada. Association for Computational Linguistics.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

NTUA-ISLab at SemEval-2019 Task 3: Determining emotions in contextual conversations with deep learning

Rolandos Alexandros Potamias, Georgios Siolas
Intelligent Systems Laboratory,
National Technical University of Athens,
Zografou, Athens , Greece
rolpotamias@gmail.com , gsiolas@islab.ntua.gr

Abstract

Sentiment analysis (SA) in texts is a well-studied Natural Language Processing task, which in nowadays gains popularity due to the explosion of social media, and the subsequent accumulation of huge amounts of related data. However, capturing emotional states and the sentiment polarity of written excerpts requires knowledge on the events triggering them. Towards this goal, we present a computational end-to-end context-aware SA methodology, which was competed in the context of the SemEval-2019 / EmoContext task (Task 3). The proposed system is founded on the combination of two neural architectures, a deep recurrent neural network, structured by an attentive Bidirectional LSTM, and a deep dense network (DNN). The system achieved 0.745 micro f1-score, and ranked 26/165 (top 20%) teams among the official task submissions.

1 Introduction and Related Work

One of the most challenging fields of Natural Language Processing (NLP) and Computational Linguistics is Sentiment Analysis (SA) that aims towards the automated extraction of a writer's sentiment or emotion as conveyed in text excerpts (Liu, 2015). Relevant efforts focus on tracking the sentiment polarity of single utterances, which in most of the cases is loaded with a lot of subjectivity and a degree of vagueness (Thelwall et al., 2010). Contemporary research in the field utilizes data from social media resources (e.g., Facebook, Twitter) as well as other short text references. However, users of social media tend to violate common grammar and vocabulary rules and even use various figurative language forms to communicate their message. In such situations, the sentiment polarity underlying the literal content of the conveyed concept may significantly differ from its figurative context, making SA tasks even more puzzling (Patra et al., 2016). Evidently, single turn text lacks in detecting sentiment polarity on sarcastic and ironic expressions (Potamias et al., 2019), as already indicatd in the relevant SemEval-2014 Task-9 *Sentiment Analysis in Twitter* (Sara et al., 2014). As sentiment reflects the emotion behind customer engagement, SA finds its realization in automated customer aware services (Kurniawati et al., 2013). A lot of research has already been devoted on capturing the sentiment polarity of contextual conversations of various utterances. Most of the relevant studies utilize single turn texts from topic related sources (e.g., Twitter). Hand-crafted and sentiment-oriented features, indicative of emotion polarity, are utilized to represent respective excerpt cases. The formed data were used as input to various traditional machine learning classifiers (e.g. SVM, Random Forests etc.) or deep learning architectures (e.g. recurrent neural networks, CNNs) in order to induce analytical models that are able to capture the underlying sentiment content of passages (Singh et al., 2018; Jianqiang et al., 2018; Hangya and Farkas, 2017).

The work presented in this study considers the recognition of the three fundamental emotions, Happy, Sad and Angry, specified in the SemEval-2019 / EmoContext task (Task 3), as a context sensitive task. In order to capture emotional categories, we settled two-layered bidirectional LSTM units that share weights over three embedded utterances resembling a siamese like architecture (Mueller and Thyagarajan, 2016). Pretrained word embeddings were utilized, Standford GloVe (Pennington et al., 2014), in order to represent single turn text input, resulting into an attentive and context-aware model. We also extended word embeddings with appropriate handling of emojis, utilizing the pretrained emoji2vec vectors (Eisner et al., 2016), and sentiment lexicons, NCR and DepecheMood (Mohammad and Turney, 2013), resulting into an enhanced representation of single turn text cases. The overall complex presents a combined neural architecture to serve SA tasks.

2 Experimental Setup: Data & Preprocessing

The SemEval-2019 / EmoContext shared task (Task 3) targets the emotion classification of user utterances in three classes, namely Happy, Sad, Angry and Other (Chatterjee et al., 2019). It provides a 33K training textual dialogue dataset in the form of three contextual turns. The distribution of data demands the elaboration of techniques that are able to cope with class imbalances in order to capture correctly the less frequent emotions, i.e., Happy, Sad and Angry (13%, 17%, 17% and 53% for Happy, Sad, Angry and Other, respectively). To handle imbalance among classes we applied a penalty weight to the loss function, proportional to the respective class frequencies.

Given that the provided data do not contain any Twitter-specific informative sign, such as hashtags and user mentions, we tried to keep the preprocessing step as simple as possible. Thus, we replaced repeated emojis and punctuation with single ones, and substituted slang abbreviations to their full expression (e.g. “bcz” is substituted by “because”).

3 System Overview

The proposed emotion analysis methodology is composed by (i) the formulation of the suitable representation schemes for the input data, and (ii) the implementation of an elaborative deep neural network architecture to map these schemes to their associated labels and the appropriate neural layers.

3.1 Embedding Layer

Word Embeddings tend to become a necessary component of deep learning approaches, with the mapping of single dimensional words to their dense vector encodings to be a critical part of the job (Mikolov et al., 2013). Vector representations are exhaustively trained on large corpora to capture the semantic content of each word. One of the most utilized vector representation is offered by Standford GloVe pretrained word vectors. However, GloVe vectors do not handle one of the most important factors in sentiment analysis, the emojis. To expand their capabilities, we append GloVe embeddings with pretrained emoji2vec, as proposed by Eisner et al. (2016). In addition, we utilized 23 extra features to enhance our pretrained word embeddings. Specifically, we elaborated 13 mood-oriented emotions, provided by the DepecheMood lexicon, as proposed by Staiano and Guerini (2014), in order to capture words mood intentions, as well as 10 NRC emotion relation scores. The utility of emotion scores is manifested in various studies (Mohammad and Turney, 2013; Kiritchenko et al., 2014). The enhancement led to a dense 323 dimensions vector representation for each word (300d-GloVe + 13d-DepecheMood emotions + 10d-NRC sentiment scores), which after re-normalization fed a recurrent neural network.

3.2 Bidirectional LSTMs and Siamese Network Architecture

In the recent years, deep learning models and in particular convolutional neural network (CNN) architectures, have become a popular and a favorable choice for several artificial intelligence tasks. However, the recurrent nature of textual data implies the need for architectures that are able to capture data of sequential nature information, which CNNs are unable to manage. To cope with the recurrent nature of textual data as well as with the sentiment contradictions that may occur in both text directions we utilize a bidirectional LSTM network architecture. In particular, we used three bidirectional LSTMs (Hochreiter and Schmidhuber, 1997) in a siamese like architecture (Bromley et al., 1994) in order to map all turns into the same vector space (Figure 1). To determine the sentiment impact of the last utterance, given the previous two, we introduce bidirectional LSTM hidden states, for each time-step and utterance. The hidden states are calculated using the same weights for each input.

3.3 Attention layer

To focus on the most significant time-sample, an attention layer (Bahdanau et al., 2014) is added on top of the regular LSTMs in order to capture and assign an importance attention factor to the hidden states, forming the so-called attentive vector $\vec{s}$:

$$r_t = tanh(W_h h_t + b_t) \quad (1)$$

$$a_t = \frac{e^{r_t}}{\sum_{j=0}^{T} e^{r_t}}, \quad \sum_{t=1}^{n} a_t = 1 \quad (2)$$

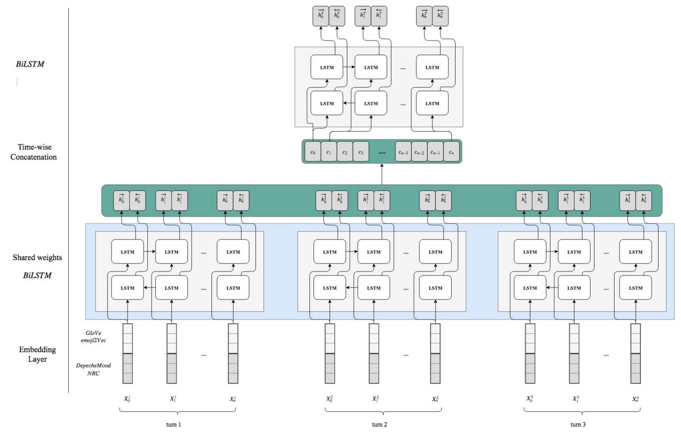


Figure 1: Siamese alike architecture, containing two layers of bidirectional LSTMs.

$$\vec{s} = \sum_{t=0}^{T} a_t h_t \tag{3}$$

where W_h and b_t are the LSTM model weights, to be optimized during training.

3.4 Dense network

To unfold and map the devised feature set we implemented a four-layered deep dense neural network, equiped with unigram and bigram Tf-idf weights of each utterance. Each neuron is activated by a ReLu function, and the final layer is concatenated with the attentive vector as defined in sub-section 3.3.

3.5 Proposed method

As already mentioned, we utilize two different and combined schemes to represent and train processed data, (i) an embedded matrix that feeds the Embeddings layer (as described in sub-section 3.1), and (ii) a uni/bi-gram Tf-idf training schema to be processed by the a layered dense DNN. In addition, the Embedding layer is connected with a siamese like bidirectional LSTM architecture, containing 164 units each. As shown in Figure 2, on top of the shared weighted LSTMs we add another bidirectional LSTM layer, also containing 164 units. To boost LSTM performance, we apply an attention mechanism on top of it, concluding to an attentive vector, as described in sub-section 3.3. We will refer to this subnetwork as *S-LSTM*, which includes the siamese like layers extended with an attentive mechanism. The Tf-idf features, as extracted for each conversation turn, are mapped onto two-layered dense neural networks containing 84 neurons each, with a ReLu non-linear activation function. The output of these layers is then concatenated and followed by another dense 84-neuron layer, creating a vector $\vec{d}$. We refer to this dense sub-network as *F-DNN*. The output vectors, $\vec{s}$ and $\vec{d}$, from the respective *S-LSTM* and *F-DNN* sub-networks are then concatenated and feed a final softmax activated dense network. The whole setup presents a combination of different and heterogeneous deep learning models.

3.6 Training

To train our model we adopted several regularization techniques to prevent overfitting. Therefore, we applied Dropout (Srivastava et al., 2014) to randomly deactivate neurons during forward training pass. We empirically set dropout parameters to 0.3 for every model layer, as well as recurrent connections of LSTM units (Gal and Ghahramani, 2016). In addition, we utilized L_2 regularization penalty loss function to every LSTM unit exceeding weight limits. Finally, we apply early-stopping technique to terminate training when loss on the development phase stop decreasing. To optimize our network we adopted Adam optimizer (Kingma and Ba, 2014) using cross entropy loss.

4 Results

Our proposed system achieved a micro f1-score ($\mathbf{f1}_{\mu}$) of 0.743, ranked the 26th in the SemEval 2019 EmoContext task. Results are presented in Table 1 and compared with different approaches. EmoContext organisers proposed a baseline classifier (referred as Baseline) that exhibits a f1-score of 0.587(Chatterjee et al., 2019). In Table 1, we compare the proposed method with the *S-LSTM* and *F-DNN* implementations, described in 3.4 and 3.5, respectively. Moreover, we present results for the SS-BED system, proposed by Gupta et al.

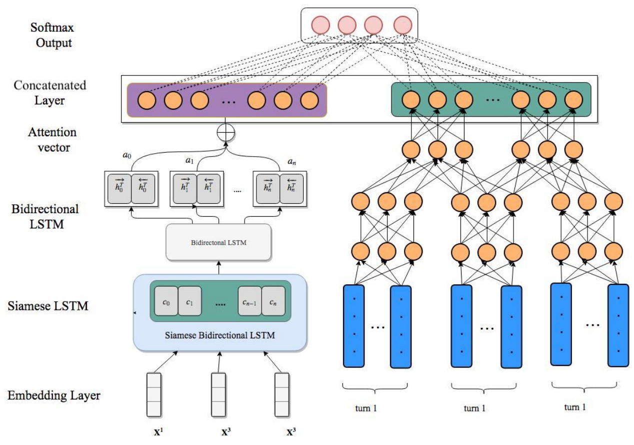


Figure 2: **Proposed Method**. the left model (S-LSTM) is composed by two LSTM layers followed by an attentive mechanism; the right model (F-DNN) is composed by two dense layers for each utterance followed by two additional dense layers

(2017). Compared with the other approaches, the proposed method exhibits a significantly higher f1-score for the Happy and Angry classes, 0.71 and 0.75, respectively. SS-BED system achieves a better performance for the Sad class (0.81) but it exhibits a poor performance for the Happy class (0.59).

To assess the importance of emoji embeddings and of the introduced mood and emotion feature sets which appended GloVe embeddings (G), we conducted additional experiments retaning the same siamese neural architecture (S-LSTM). In each experiment we extended GloVe embeddings with a respective feature set, i.e., the emoji2vec embeddings (E), the 13 DepecheMood (D) mood intensions, and the 10 NRC emotion relations (N). Compared to the S-LSTM$_G$ neural classifier, the use of additional embedding sets improve slightly the $f1_\mu$ performance, with the utilization of the emoji embeddings to achieve the highest increase (S-LSTM$_{G+E}$, from 0.67 to 0.70). But their yield remains lower compared to the respective architecture where all embedding sets are utilised (S-LSTM$_{all}$, 0.72).

In summary, the results demonstrate the superiority of the proposed method over all other approaches, and signifies its ability to succeed stable results over the different sentiment classes.

5 Conclusion

In this study we implemented a combination of two different representations and respective training schemes for the input data. First, we extended pretrained GloVe embedding vectors with emojis and appended 23 additional emotional features. In addition, we developed an S-LSTM model, containing a siamese alike bidirectional LSTM architecture with its output to feed another bidirectional LSTM layer followed by an attention layer. Furthermore, we transformed input data by their Tf-idf weight representations in order to feed a dense deep neural network (F-DNN).

System	**f1**$_{Happy}$	**f1**$_{Sad}$	**f1**$_{Angry}$	**f1**$_\mu$
Baseline	-	-	-	0.58
SS-BED	0.59	**0.81**	0.74	0.71
F-DNN	0.70	0.68	0.73	0.70
S-LSTM$_{all}$	0.69	0.76	0.71	0.72
S-LSTM$_G$	0.67	0.69	0.65	0.67
S-LSTM$_{G+E}$	0.65	0.71	0.74	0.70
S-LSTM$_{G+D}$	0.68	0.68	0.71	0.69
S-LSTM$_{G+N}$	0.66	0.71	0.68	0.68
Proposed	**0.71**	0.77	**0.75**	**0.74**

Table 1: Comparison results: Proposed method vs. other approaches (**f1**$_\mu$ refer to the f1-micro metric)

All model features are appropriately mapped and concatenated in order to feed the final dense softmax layer. Comparative results demonstrate the superiority of the proposed method over other approaches and single network models, as well as its robustness with regard the stability of performance over different emotional classes. Thus we could state that the proposed methodology defines an end-to-end solution on sentiment analysis and classification tasks, suited for imbalanced data, with the ability as well, to cope with huge amounts of data.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. *arXiv preprint arXiv:1609.08359*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations. *CoRR*, abs/1707.06996.

Viktor Hangya and Richárd Farkas. 2017. A comparative empirical study on social media sentiment analysis over various genres and languages. *Artificial Intelligence Review*, 47(4):485–505.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780.

Zhao Jianqiang, Gui Xiaolin, and Zhang Xuejun. 2018. Deep convolution neural networks for twitter sentiment analysis. *IEEE Access*, 6:23253–23260.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif Mohammad. 2014. Nrc-canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 437–442.

Kurniawati Kurniawati, Graeme G Shanks, and Nargiza Bekmamedova. 2013. The business impact of social media analytics. In *ECIS*, volume 13, page 13.

Bing Liu. 2015. *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge University Press.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Saif M Mohammad and Peter D Turney. 2013. Nrc emotion lexicon. *National Research Council, Canada*.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2786–2792. AAAI Press.

Braja Gopal Patra, Soumadeep Mazumdar, Dipankar Das, Paolo Rosso, and Sivaji Bandyopadhyay. 2016. A multilevel approach to sentiment analysis of figurative language in twitter. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 281–291. Springer.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*, volume 14, pages 1532–1543.

Rolandos Alexandros Potamias, Georgios Siolas, and Andreas Stafylopatis. 2019. A robust deep ensemble classifier for figurative language detection. In *20th International Conference on Engineering Applications of Neural Networks*. Springer.

Rosenthal Sara, Ritter Alan, Nakov Preslav, and Stoyanov Veselin. 2014. Semeval-2014 task 9: Sentiment analysis in twitter. In *Proc. of the 8th International Workshop on Semantic Evaluation*, pages 73–80.

Nikhil Kumar Singh, Deepak Singh Tomar, and Arun Kumar Sangaiah. 2018. Sentiment analysis: a review and comparative analysis over social media. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–21.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Jacopo Staiano and Marco Guerini. 2014. Depechemood: a lexicon for emotion analysis from crowd-annotated news. *arXiv preprint arXiv:1405.1605*.

Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 61(12):2544–2558.

ntuer at SemEval-2019 Task 3: Emotion Classification with Word and Sentence Representations in RCNN

Peixiang Zhong, Chunyan Miao
Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly
Nanyang Technological University Singapore
peixiang001@e.ntu.edu.sg, ascymiao@ntu.edu.sg

Abstract

In this paper we present our model on the task of emotion detection in textual conversations in SemEval-2019. Our model extends the Recurrent Convolutional Neural Network (RCNN) by using external fine-tuned word representations and DeepMoji sentence representations. We also explored several other competitive pre-trained word and sentence representations including ELMo, BERT and InferSent but found inferior performance. In addition, we conducted extensive sensitivity analysis, which empirically shows that our model is relatively robust to hyper-parameters. Our model requires no handcrafted features or emotion lexicons but achieved good performance with a micro-F1 score of 0.7463.

1 Introduction

Emotions are psychological and physiological states generated in humans in reaction to internal or external events. Messages in human conversations inherently convey emotions. With the rise of social media platforms such as Twitter, as well as chatbots such as Amazon Alexa, there is an emerging need for machines to understand human emotions in conversations, which has a wide range of applications such as opinion analysis in customer support (Devillers et al., 2002) and providing emotion-aware responses (Zhong et al., 2019). SemEval-2019 Task 3: EmoContext (Chatterjee et al., 2019b) is designed to promote research in this task.

This task is to detect emotions in textual conversations. Each conversation is composed of three turns of utterances and the objective is to detect the emotion of the last utterance given the first two utterances as the context. The emotions in this classification task include happy, sad, angry and others, adapted from the well-known Ekman's six basic emotions: anger, disgust, fear, happiness, sadness, and surprise (Ekman, 1992). The evaluation criteria is micro-averaged F1 score since the data is extremely unbalanced, as shown in Table 1.

In recent years, pre-trained word and sentence representations achieved very competitive performance in many NLP tasks, e.g., fine-tuned word embeddings using distant training (Cliche, 2017) and tweet sentence representations DeepMoji (Felbo et al., 2017) on sentiment analysis, and contextualized word representations BERT (Devlin et al., 2018) on 11 NLP tasks. Motivated by these successes, in this task we explored different word and sentence representations. We then fed these representations into a Recurrent Convolutional Neural Network (RCNN) (Lai et al., 2015) for classification. RCNN includes a Long short-term memory (LSTM) network (Hochreiter and Schmidhuber, 1997) to capture word ordering information and a max-pooling layer (Scherer et al., 2010) to learn discriminative features. We also experimented LSTM and CNN in our preliminary analysis but achieved worse performance as compared to RCNN. Our final system adopted fine-tuned word embeddings and DeepMoji as our choices of word and sentence representations, respectively, due to their superior performance on the validation dataset. The code is publicly available at Github[1].

2 Related Work

Emotion detection in textual conversations is an under-explored research task. The majority of existing works focused on the multi-modality settings (Devillers et al., 2002; Hazarika et al., 2018; Majumder et al., 2019). Chatterjee et al. (2019a) is one of the early works on the textual modality that first collected the dataset used in this task and then

[1]https://github.com/zhongpeixiang/SemEval2019-Task3-EmotionDetection

Dataset Split	Size	#Happy	#Sad	#Angry	#Others	Average Utterance Length
Train	30160	4243	5463	5506	14948	5.22
Val	2755	142	125	150	2338	5.05
Test	5509	284	250	298	4677	5.05

Table 1: Total number of conversations and their distributions over each emotion class for each dataset split. Average number of tokens per utterance for each dataset split are also reported.

proposed an LSTM model with both semantic and sentiment embeddings to classify emotions. This task is also closely related to sentiment analysis (Pang et al., 2008) where the opinions of a piece of text is to be identified. One major difference between them is that this task detects emotions only on the last portion of a piece of text and the rest is treated as context.

Our model leverages pre-trained word and sentence representations. There is a research trend on word and sentence embeddings after the invention of Word2Vec (Mikolov et al., 2013). Cliche (2017) fine-tuned word embeddings using CNN-based sentiment classification model and distant training (Go et al., 2009). Peters et al. (2018) proposed a contextualized word embedding named ELMo to incorporate context information and solve the polysemy issues in conventional word embeddings. Devlin et al. (2018) proposed another contextualized word embedding named BERT by extending the context to both directions and training on the masked language modelling task. Kiros et al. (2015) proposed a sentence-level representation named SkipThought, which shares similar ideas to Word2Vec but operates on sentence level. Conneau et al. (2017) proposed InferSent by learning sentence representations on natural language inference tasks. Felbo et al. (2017) proposed DeepMoji by learning tweet sentence representations in the emoji classification task using 1246 million tweets and distant training.

Our RCNN model is closely related to neural network based sentiment analysis models. Two of the most popular models are LSTMs and CNNs. LSTM-based models can capture the word ordering information and have achieved the state-of-the-art performance on many sentiment analysis datasets (Gray et al., 2017; Liu et al., 2018; Howard and Ruder, 2018). CNN-based models can capture local dependencies, discriminative features, and are parallelizable for efficient computation (Kim, 2014; Johnson and Zhang, 2017).

3 System Description

In this section we describe our data preprocessing procedures and illustrate how we leverage pre-trained word and sentence representations in our RCNN model. The overall architecture is depicted in Figure 1.

3.1 Data Preprocessing

We concatenated three utterances as one sentence, separated by EOS tokens. We used the tokenizer from Spacy[2] for tokenization. We removed training sentences that have more than 75 tokens. We removed duplicate punctuations and spaces. We kept all remaining tokens in the training dataset as the vocabulary.

3.2 Pre-trained Word Representation

We fine-tuned the word embeddings obtained from (Baziotis et al., 2017), which has an embedding size of 100 and is pre-trained on 330M English Twitter messages using Glove (Pennington et al., 2014). The fine-tuning is conducted on the binary sentiment classification task using the basic CNN model (Kim, 2014) on 1.6 million tweets (Go et al., 2009). These tweets are labelled with positive and negative sentiments. Fine-tuning on these tweets introduces sentiment-discriminative features to word embeddings (Cliche, 2017). The CNN model has kernel sizes of 1, 2, and 3. Each kernel size has 300 filters. During fine-tuning, the embedding layer is first frozen for one epoch and then unfrozen for another three epochs.

3.3 Pre-trained Sentence Representation

We adopted DeepMoji (Felbo et al., 2017) as the sentence representations in our model. Each sentence will be encoded into a vector of size 2304. DeepMoji is trained on the 64-class emoji classification task using 1246 million tweets. Since emoji reflects emotions and sentiments, DeepMoji is an ideal model to provide emotion-discriminative sentence representations. We also

[2]https://spacy.io/

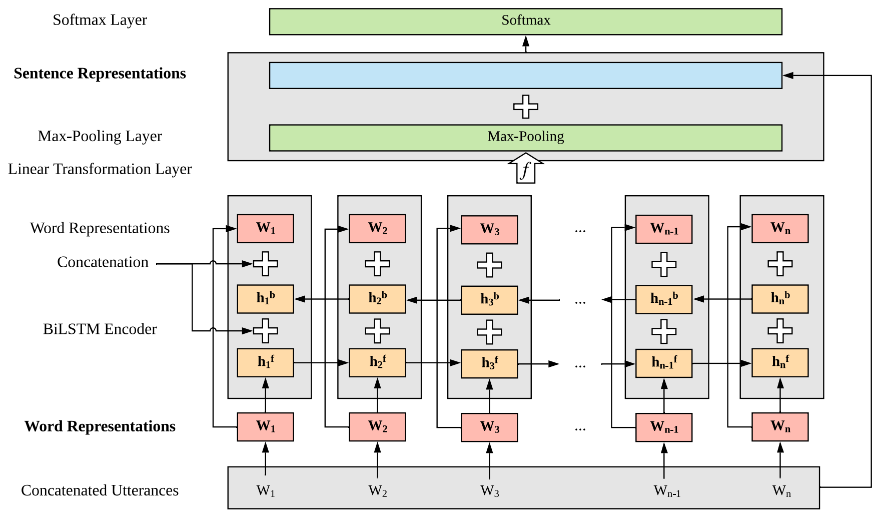


Figure 1: Overall architecture of our proposed model.

explored InferSent (Conneau et al., 2017), another sentence representation model with competitive performance on sentence classification and information retrieval tasks (Perone et al., 2018).

3.4 RCNN

As shown in Figure 1, we fed word and sentence representations into a RCNN model. The RCNN model mainly comprises of a two-layer Bidirectional LSTM (BiLSTM), a linear transformation layer and a max-pooling layer. At the embedding layer, each sentence is transformed to a sequence of word embeddings W_i of size 100 using our pre-trained word representations, where $i = 1, 2, ..., n$, and n is the number of tokens in the concatenated utterance. The BiLSTM encodes these word embeddings into hidden states h_i^f, h_i^b in both forward and backward directions, respectively, where each direction has a hidden size of 200. The hidden states in both directions are concatenated together, along with the word representations W_i to form a vector of size 500. A linear transformation is then applied to project the resulted vector into a vector of size 200, followed by a max-pooling layer to extract discriminative sentence features. Finally, the DeepMoji sentence representation is concatenated with the pooled vector to form a final sentence representation of size 2504, followed by a softmax layer for classification.

3.5 Training

We train our model on the training dataset and fine-tune on the validation dataset based on the micro-F1 score. Since the dataset is highly unbalanced, we use weighted cross-entropy loss for optimization, where the weights are the ratio of validation dataset label distribution to training dataset label distribution, followed by a normalization to ensure that the sum of weights is 1. We use Adam (Kingma and Ba, 2014) optimizer with a learning rate of 0.0005 and batch size of 64. We clip the norm of gradients to 5. We trained our model 6 epochs. The learning rate is annealed by a factor of 0.2 every epoch after epoch 5. We also freeze the embedding for the first two epochs. We use dropout rate of 0.5 in BiLSTM and 0.7 in linear layers. The model is implemented in PyTorch.

4 Result Analysis

In this section we explored different word and sentence representations and compared their performance on the test set. We also conducted sensitivity analysis for our model hyper-parameters. All results are averaged across 5 different seeds. It is worth noting that the settings with the best test scores in the analysis below are not exactly the same as our best system on the leaderboard since our best system is fine-tuned on the validation dataset, which do not guarantee to produce the best test results.

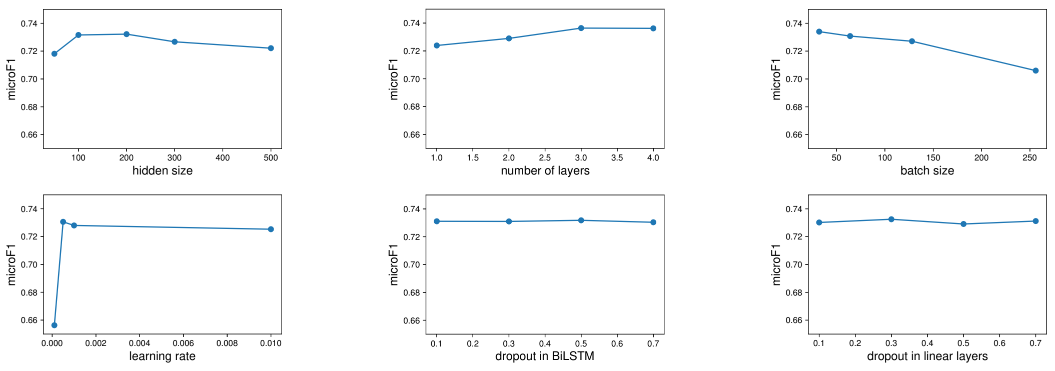


Figure 2: Sensitivity analysis on model hyper-parameters.

Word Representation	micro-F1
Original Glove	0.7250
Pre-trained Glove	0.7279
Fine-tuned Glove	0.7339
ELMo	0.6990
BERT	0.6656

Table 2: Micro-F1 score on the test set using different word representations.

Sentence Representation	micro-F1
None	0.7194
InferSent (GloVe)	0.7259
InferSent (fastText)	0.7277
DeepMoji	0.7299
DeepMoji + InferSent (GloVe)	0.7298
DeepMoji + InferSent (fastText)	0.7344

Table 3: Micro-F1 score on the test set using different sentence representations.

We explored the original GloVe embedding trained on 27B tweet tokens[3], pre-trained GloVe embedding[4], our fine-tuned GloVe embedding, ELMo embedding and BERT embedding. The results are shown in Table 2. Fine-tuned GloVe embedding performs noticeably better than the original GloVe embedding and the pre-trained GloVe embedding. Surprisingly, contextualized embeddings such as ELMo and BERT perform worse than the original GloVe embedding. Possible reasons for their inferior performance are 1) they are fixed during training, which may hinder the overall optimization. 2) they have large embedding size, which can easily cause overfitting.

We explored no sentence embedding, InferSent trained on GloVe, InferSent trained on fastText, and DeepMoji. The results are shown in Table 3. It is clear that sentence representations improved model performance significantly. In particular, DeepMoji achieves the best performance for single sentence representation. InferSent trained on fastText consistently performs better than InferSent trained on GloVe. In addition, concatenating two sentence representations together further improved model performance.

We conducted sensitivity analysis on our model hyper-parameters: hidden size, number of layers in BiLSTM, batch size, learning rate, dropout in BiLSTM and dropout in linear layers. The results are depicted in Figure 2. Our model is relatively robust to hyper-parameters except for the learning rate. When learning rate is around 0.0001 or smaller, the model is unable to be trained effectively.

5 Conclusion

In this paper we presented our model on the task of emotion detection in textual conversations in SemEval-2019. We explored different word and sentence representations in the RCNN model and achieved competitive results. Our result analysis indicate that both pre-trained word and sentence representations help improve the performance of RCNN. However, currently popular contextualized word representations such as ELMo and BERT produced inferior results.

Acknowledgement

This research is supported, in part, by the National Research Foundation, Prime Minister's Office, Singapore under its IDM Futures Funding Initiative and the Singapore Ministry of Health under its National Innovation Challenge on Active and Confident Ageing (NIC Project No. MOH/NIC/HAIG03/2017).

[3]https://nlp.stanford.edu/projects/glove/

[4]https://github.com/cbaziotis/datastories-semeval2017-task4

References

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *SemEval-2017*, pages 747–754, Vancouver, Canada.

Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. 2019a. Understanding emotions in text using deep learning and big data. *Computers in Human Behavior*, 93:309–317.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019b. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *SemEval-2019*, Minneapolis, Minnesota.

Mathieu Cliche. 2017. Bb_twtr at semeval-2017 task 4: Twitter sentiment analysis with cnns and lstms. *arXiv preprint arXiv:1704.06125*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*, pages 670–680, Copenhagen, Denmark.

Laurence Devillers, Ioana Vasilescu, and Lori Lamel. 2002. Annotation and detection of emotion in a task-oriented human-human dialog corpus. In *proceedings of ISLE Workshop*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Paul Ekman. 1992. An argument for basic emotions. *Cognition & emotion*, 6(3-4):169–200.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *EMNLP*, pages 1615–1625.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12).

Scott Gray, Alec Radford, and Diederik P Kingma. 2017. Gpu kernels for block-sparse weights. Technical report, Technical report, OpenAI.

Devamanyu Hazarika, Soujanya Poria, Amir Zadeh, Erik Cambria, Louis-Philippe Morency, and Roger Zimmermann. 2018. Conversational memory network for emotion recognition in dyadic dialogue videos. In *NAACL*, volume 1, pages 2122–2132.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*, volume 1, pages 328–339.

Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *ACL*, volume 1, pages 562–570.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*, pages 3294–3302.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273.

Fei Liu, Trevor Cohn, and Timothy Baldwin. 2018. Recurrent entity networks with delayed memory update for targeted aspect-based sentiment analysis. In *NAACL*, volume 2, pages 278–283.

Navonil Majumder, Soujanya Poria, Devamanyu Hazarika, Rada Mihalcea, Alexander Gelbukh, and Erik Cambria. 2019. Dialoguernn: An attentive rnn for emotion detection in conversations. In *AAAI*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Christian S Perone, Roberto Silveira, and Thomas S Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv preprint arXiv:1806.06259*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*, volume 1, pages 2227–2237.

Dominik Scherer, Andreas Müller, and Sven Behnke. 2010. Evaluation of pooling operations in convolutional architectures for object recognition. In *ICANN*, pages 92–101. Springer.

Peixiang Zhong, Di Wang, and Chunyan Miao. 2019. An affect-rich neural conversational model with biased attention and weighted cross-entropy loss. In *AAAI*.

PKUSE at SemEval-2019 Task 3: Emotion Detection with Emotion-Oriented Neural Attention Network

Luyao Ma[1,2*], **Long Zhang**[1,2*], **Wei Ye**[1], **Wenhui Hu**[1]
[1]National Engineering Research Center for Software Engineering, Peking University
[2]School of Software and Microeconomics, Peking University
{maluyao, zhanglong418, wye, huwenhui}@pku.edu.cn

Abstract

This paper presents the system in SemEval-2019 Task 3, "EmoContext: Contextual Emotion Detection in Text". We propose a deep learning architecture with bidirectional LSTM networks, augmented with an emotion-oriented attention network that is capable of extracting emotion information from an utterance. Experimental results show that our model outperforms its variants and the baseline. Overall, this system has achieved 75.57% for the microaveraged F1 score.

1 Introduction

With the rapid development of social media platforms like Twitter, a huge number of textual dialogues has increasingly emerged. It is a challenge for chat bots to generate responses based on user emotions which can avoid inappropriate conversations. Emotion detection in text (Chatterjee et al., 2019) is a research area within Natural Language Processing which is aim to detect the emotion of user expressed in text.

Many techniques have been proposed, Wang et al.,Hasan et al.,Liew and Turtle used feature engineering to extract features manually. In this area, deep learning-based approches have performed well in recent years. Some methods (Wöllmer et al., 2010; Metallinou et al., 2012; Poria et al., 2017; Chernykh et al., 2017) used recurrent neural network to model the sequence of utterances for emotion detection. However, those models did not highlight the emotion-related parts. We use attention mechanism to locate the parts expressing emotions in the utterance.

The Task3 in Semeval-2019 is to detect contextual emotions in text. For this task, we propose a deep learning approach which is a combination of Long Short-Term Memory network and attention mechanism.

The rest of the paper is organized as follows: Section 2 provides system overview. Section 3 describes our approach in detail. Our experiment is discussed in Section 4. We conclude our work in Section 5.

*These authors have contributed equally to this work.

2 System Overview

2.1 Text Preprocessing and Word Embedding

We use word embeddings as input to the model. Word embeddings are distributed vector presentations of words (Mikolov et al., 2013), capturing their syntactic and semantic information. A good word embedding can get a better classification performance. After comparison, we find that the effect of the GloVe (Pennington et al., 2014) is the best, but when we turn the word into a word vector, we find a lot of cases that are out of vocabulary(oov). In view of that, we preprocess the data as follows:

- The emoji used in the chat can better express human emotions, so we turn them into corresponding emotion words and add them to the sentence, which not only solves the oov, but also increases the emotion information in the sentence
- Several emoticons are replaced by the tokens "happy", "sad", "angry"
- All words are lowercased

2.2 Long Short-Term Memory

LSTM is a special form of threshold RNN (Hochreiter and Schmidhuber, 1997), which is designed to deal with sequential data by sharing its internal weights across the sequence. Different from the structure of RNN, LSTM has three gates:

an input gate i_t, a forget gate f_t, an output gate o_t and a memory cell c_t. Their effect is to allow the network to store and retrieve information over long periods of time.

In our approach, we use the bidirectional LSTM model to better capture the contextual information in sentences. Schuster and Paliwal shows that the bidirectional structure has better performance in classification experiments. In order to better handle the relations among the utterances of a dialogue, we use the bc-LSTM architecture(Poria et al., 2017) to process the dialogue-level classification. The architecture preserves the sequential order of utterances when constructing the dialogue representation.

2.3 Attention Mechanism

The attention mechanism was originally applied to image recognition (Itti and Koch, 2001; Mnih et al., 2014), mimicking the focus of the eye moving on different objects when the person viewed the image. Similarly, when people read an article, their attention to each part of the text is different. The attention mechanism imitates human behavior, giving each feature different weights. With the weight of a feature being greater, the contribution of this to current recognition becomes greater. Neural networks with attention mechanism have been applied in many tasks of NLP, including machine translation (Bahdanau et al., 2014; Luong et al., 2015) text summarization (Rush et al., 2015) text classification (Yang et al., 2016) sentiment classification (Chen et al., 2016) and stance classification (Du et al., 2017).

When learning the representations of text sequences, word embeddings are the most effective intermediate representations for capturing semantic information. We embed the classification label and word into the same semantic space, and then construct the semantic relatedness of them according to the similarity of word embeddings. Our model obtains the attention weights of the words through the emotion-oriented attention network, which highlights the emotion words, thus improving the performance of the emotion classification.

3 Model Description

Our model has two steps as follows: 1. Extract the features of each utterance in the dialogue 2. Construct the representation of the dialogue by the features of three utterances for emotion classification.

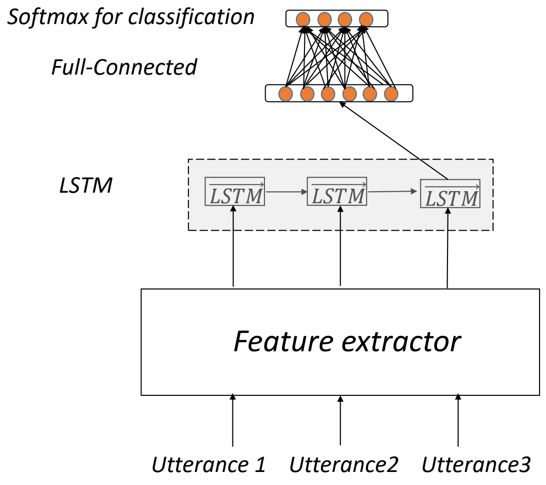


Figure 1: Architecture for emotion classification.

In the feature extraction step: the embedding of each utterance is fed into the BiLSTM layer to construct the word representation of each word; meanwhile we obtain the attention weight of the corresponding word by the emotion-oriented attention network. We use the inner product of them to represent the word, and then feed it into the BiLSTM layer. Finally, we get the representation of each utterance after the pooling operation(Fig. 2).

In the classification step: the features of the three utterances obtained from the previous step are fed into the LSTM layer as timing information for emotion classification(Fig. 1).

3.1 Embedding Layer

An input sequence $\boldsymbol{X}$ of length T is composed of word tokens: $\boldsymbol{X} = \{x_1, ..., x_T\}$. Each token x_t is replaced with the corresponding vocabulary index $V(t)$. The embedding layer transforms the token into vector $e_t \in \mathbb{R}^d$ which is selected from the embedding matrix E according to the index, where d is the dimensionality of the embedding space.

In order to highlight the emotion words in the sequence, we append the word embedding vector of "emotion" to the embedding of each word in original text. The emotion-augmented embedding of a word t is the concatenation of the embedding vector e_t and the emotion representation e_z,

$$e_t^z = e_t \| e_z \qquad (1)$$

where $\|$ denotes the concatenation operation, and then the dimention of e_t^z is $2d$.

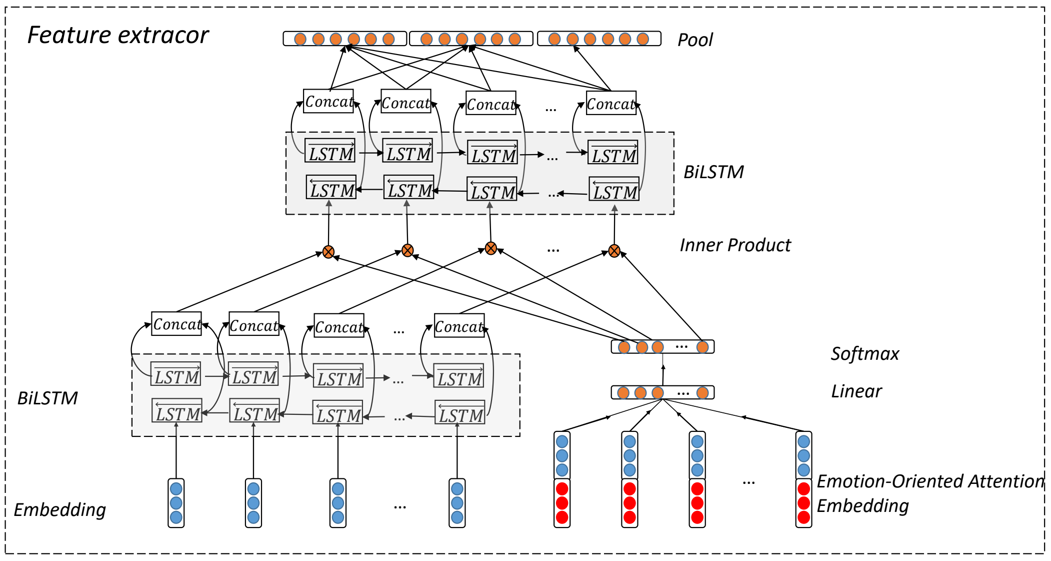


Figure 2: Feature extractor of an utterance with emotion-oriented attention network.

3.2 BiLSTM Layer

The LSTM reads the sequence $\boldsymbol{X}$ only in one direction. We use a bidirectional LSTM to get annotations of words by summarizing the contextual information from both directions. A bidirectional LSTM consists of a forward LSTM $\overrightarrow{f}$ that reads the sentence from x_1 to x_T and a backward LSTM $\overleftarrow{f}$ that reads the sentence from x_T to x_1. We obtain the annotation h_t for each word x_t, by concatenating the forward hidden state $\overrightarrow{h}_t$ and the backward one $\overleftarrow{h}_t$,

$$h_t = \overrightarrow{h}_t \| \overleftarrow{h}_t \tag{2}$$

3.3 Emotion-Oriented Attention Network

In the task, the emotion words in the conversation are vital for classification, which cannot be captured by the BiLSTM. In order to highlight the emotion-related words in the utterance, we design an attention mechanism which increases the weight of the important words on the basis of the BiLSTM and contributes more to the classification decision.

We apply a linear layer to convert the emotion-augmented embedding of a word e_t^z to a scalar value u_t, and then get a normalized importance weight α_t through a softmax function. This weight is producted with the word representation h_t to get a weighted word representation v_t for each word.

$$u_t = W_u e_t^z + b_u \tag{3}$$

$$\alpha_t = \frac{e^{u_t}}{\sum_{i=1}^{T} e^{u_i}} \tag{4}$$

$$v_t = \alpha_t h_t \tag{5}$$

3.4 Pooling Layer

From the idea of network in network, we use global maxpooling, global averagepooling and last tensor for the matrix f output of the BiLSTM layer. Maxpooling can get the most important features of all features (Scherer et al., 2010). Averagepooling can get the most common features of all features. The last tensor output $\bar{l}$ of the matrix f can obtain the semantic information of the sentence in forward and backward through BiLSTM.

The utterance representation z is obtained by the concatenation of the max vector $\overline{m}$, the average vector $\bar{a}$ and the last vector $\bar{l}$.

$$z = \overline{m} \| \bar{a} \| \bar{l} \tag{6}$$

3.5 Emotion Classification

We use the three utterance representations obtained by feature extractor shown in Figure 2 to construct the dialogue representation. The three utterance representations $[z_1, z_2, z_3]$ are fed into the LSTM, and the last time-step hidden state h_3 of the LSTM is regarded as the dialogue representation r. We pass it to a fully-connected network with a softmax activation function. This layer obtains a normalized four-dimensional vector through the nonlinear transformation function of the input vector.

$$p = \mathrm{softmax}(W_f h_3 + b_f) \tag{7}$$

where W_f and b_f are the weights and bias terms of the fully-connected layer.

4 Evaluation

4.1 Data

The datasets are provided by Semeval-2019 Task 3. Table 1 gives an overview of the datasets. All the conversations are collected from twitter. The conversations consist of user 1's tweet, user 2's response to the tweet and user 1's response to user2. The label is the emotion of the third turn that human judges mark after considering the context of three rounds of dialogue.

Dataset	Happy	Sad	Angry	Others	Total
Training	4243	5463	5506	14948	30160
Validation	425	547	551	1495	3018
Test	284	250	298	4677	5509

Table 1: Datasets for Semeval-2019 Task 3.

4.2 Experiments

The model is implemented using Keras 2.0 (Chollet et al., 2017). We experiment with Stanford's GloVe 300 dimensional word embeddings trained on 840 billion words from Common Crawl. Our model is trained with Adam Optimizer (Kingma and Ba, 2014) with initial learning rate of 0.001 and batch size of 64. We use BiLSTMs with hidden state size 256, with dropout rate 0.5 on the first BiLSTM layer and dropout rate 0.3 on the second one to prevent our neural network from overfitting (Srivastava et al., 2014).

In our task, the size of samples for each class is not balanced, which will result in the model tending to be biased toward the majority class with poor accuracy for the minority class. For this, we adjust the parameter 'class_weight' to weight the loss function of each class during training. This can be useful to tell the model to "pay more attention" to samples from an under-represented class. In this case, we set the parameter 'class_weight' (Happy : 2, Sad : 1, Angry : 1, Others : 4)

4.3 Result and Analysis

In order to evaluate the effect of the emotion-oriented attention network and the balanced class weights, we compare our approach with its variants and the baseline.

Variant1: The variant does not adjust the parameter 'class_weight'.

Variant2: The variant changes the emotion-oriented attention network with the attention machanism used in (Yang et al., 2016)

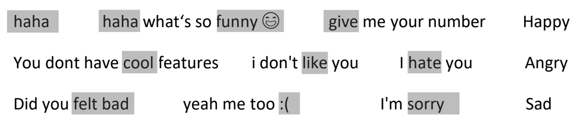
Variant3: The variant removes the emotion-oriented attention network from the model.

model	Happy	Sad	Angry	MicroF1
Baseline	0.5461	0.6149	0.5945	0.5861
Variant1	0.7082	0.7574	0.7175	0.7264
Variant2	0.6967	0.7683	0.7375	0.7330
Variant3	0.7051	**0.8113**	0.7182	0.7423
Our Model	**0.7138**	0.8088	**0.7500**	**0.7557**

Table 2: Performance of our system and its variants.

Table 2 shows that our model outperforms the other variants which are all above the baseline 0.5861 for the micro-averaged F1 score. Variant3 has the best performance on 'Sad' class and our model has the best performance on two classes and micro-averaged F1 score.

To validate that our model has the ability to capture the emotion-related parts of an utterance, we visualize the weights of attention for the following three dialogues. Figure 3 shows that the emotion words are highlighted in the dialogues, such as 'Haha', 'funny', 'cool', 'like', 'hate', 'felt', 'bad', 'SORRY', but the model also highlights some trivial words, such as 'Give'.



haha	haha what's so funny 😊	give me your number	Happy
You dont have cool features	i don't like you	I hate you	Angry
Did you felt bad	yeah me too :(	I'm sorry	Sad

Figure 3: Visualization of attention of examples.

5 Conclusion

In this paper, we proposed an emotion-oriented neural attention network for Semeval-2019 Task 3. The network use the attention mechanism to select the emotion-related parts in the utterances. The classification performance of our model is better than its variants and the baseline. Meanwhile, the visualization shows that the model has captured more decision-making information in the dialogue.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016. Neural sentiment classification with user and product attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1650–1659.

Vladimir Chernykh, Grigoriy Sterling, and Pavel Prihodko. 2017. Emotion recognition from speech with recurrent neural networks. *arXiv preprint arXiv:1701.08071*.

François Chollet et al. 2017. Keras https://github.com/fchollet/keras.

Jiachen Du, Ruifeng Xu, Yulan He, and Lin Gui. 2017. Stance classification with target-specific neural attention networks. International Joint Conferences on Artificial Intelligence.

Maryam Hasan, Emmanuel Agu, and Elke Rundensteiner. 2014. Using hashtags as labels for supervised learning of emotions in twitter messages. In *ACM SIGKDD Workshop on Health Informatics, New York, USA*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Laurent Itti and Christof Koch. 2001. Computational modelling of visual attention. *Nature reviews neuroscience*, 2(3):194.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jasy Suet Yan Liew and Howard R Turtle. 2016. Exploring fine-grained emotion detection in tweets. In *Proceedings of the NAACL Student Research Workshop*, pages 73–80.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Angeliki Metallinou, Martin Wollmer, Athanasios Katsamanis, Florian Eyben, Bjorn Schuller, and Shrikanth Narayanan. 2012. Context-sensitive learning for enhanced audiovisual emotion classification. *IEEE Transactions on Affective Computing*, 3(2):184–198.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh, and Louis-Philippe Morency. 2017. Context-dependent sentiment analysis in user-generated videos. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 873–883.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Dominik Scherer, Andreas Müller, and Sven Behnke. 2010. Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks*, pages 92–101. Springer.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Wenbo Wang, Lu Chen, Krishnaprasad Thirunarayan, and Amit P Sheth. 2012. Harnessing twitter" big data" for automatic emotion identification. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 587–592. IEEE.

Martin Wöllmer, Angeliki Metallinou, Florian Eyben, Björn Schuller, and Shrikanth Narayanan. 2010. Context-sensitive multimodal emotion recognition from speech and facial expression using bidirectional lstm modeling. In *Proc. INTERSPEECH 2010, Makuhari, Japan*, pages 2362–2365.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Podlab at SemEval-2019 Task 3: The Importance of Being Shallow

Andrew Nguyen[1,2], **Tobin South**[1,2], **Nigel G. Bean**[1,2], **Jonathan Tuke**[1,2] and **Lewis Mitchell**[1,2,3]
[1]ARC Centre of Excellence in Mathematical and Statistical Frontiers
[2]School of Mathematical Sciences, The University of Adelaide
[3]Stream Leader, Data to Decisions CRC

Abstract

This paper describes our linear SVM system for emotion classification from conversational dialogue, entered in SemEval2019 Task 3. We used off-the-shelf tools coupled with feature engineering and parameter tuning to create a simple, interpretable, yet high-performing, classification model. Our system achieves a micro F1 score of 0.7357, which is 92% of the top score for the competition, demonstrating that "shallow" classification approaches can perform well when coupled with detailed feature selection and statistical analysis.

1 Introduction

Sentiment analysis from textual data is a well-studied topic, however the particular problem of inferring the emotional context of text-based *conversations* is relatively little-studied. With the increasing abundance of conversational text-based datasets, particularly from social media, there is increasing interest in studying such datasets across a wide range of domains, from unsolicited opinion polling (Cody et al., 2015) to mental health monitoring (Miner et al., 2016). Highly sophisticated systems for emotion detection on conversational data exist (Gupta et al., 2017), however these may not be appropriate for researchers in other fields, and it is desirable to develop *interpretable* models where the informative features can be readily interpreted and measures of confidence can be assigned to individual predictions.

Our aim in entering SemEval2019 therefore was to build a high-performing model using only off-the-shelf tools and some feature engineering, without resorting to deep neural network architectures or overly sophisticated approaches. This model only uses linear Support Vector Machines (SVMs) (Hastie et al., 2009) and feature engineering using the Natural Language Toolkit (NLTK) (Bird et al., 2009). These relatively "shallow" (as opposed to "deep") approaches to classification from text have been used, for example, in event prediction from social media data (Tuke et al., 2018). Despite its simplicity, our system performed very well, achieving a maximum score in the top 22% of all submissions.

2 Task

The task for Semeval 2019 Task 3 is classifying the correct emotion (`happy`, `angry`, `sad` or `others`) on a 3-turn text communication dialogue (Chatterjee et al., 2019). Table 1 shows an example conversation.

Turn 1	You are funny
Turn 2	LOL I know that :p
Turn 3	=)
Label	Happy

Table 1: A example of the 3-turn text dialogue.

The Training dataset contained 5 columns:

- ID: a unique number to identify each training sample;
- Turn 1: the first turn in the 3-turn conversation, written by User A;
- Turn 2: a reply to Turn 1 by User B;
- Turn 3: reply to Turn 2, written again by User A;
- Label: the human-judged label of emotion of Turn 3 based on the conversation for the given training sample. It is always one of the four labels: `happy`, `sad`, `angry` and `others`.

The organisers provided three datasets:

- `Training` contains 30,160 examples with the correct labels and approximately 50% `others`;
- `Dev` contains 2,755 examples with correct labels (approximately 88% `others`);
- `Test` contains 5,509 examples without labels, but participants are told that the split is approximately the same as the `Dev` set, i.e., ≈88% `others`.

This last piece of information given by the organisers regarding the balance of the `Test` set will be critial to our system described below, as we will employ a *post hoc* class balancing technique to match these proportions.

Evaluation is performed using the micro-averaged F1 score for the three emotion classes i.e. `happy`, `sad` and `angry` on the `Test` set.

3 Proposed System

Figure 1 shows an overview of our system, which we describe below in detail.

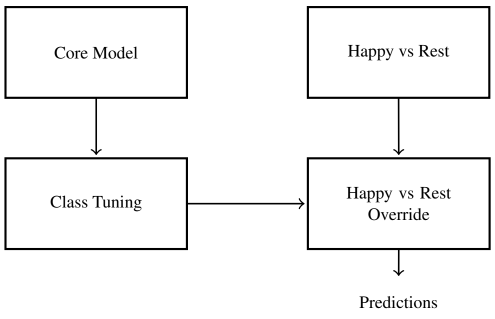


Figure 1: Model pipeline. Our core SVM model performs well for the `sad` and `angry` classes, but underestimates the `happy` class. The "Happy vs Rest" override classifier therefore is designed to achieve high accuracy on this class specifically.

3.1 Feature Engineering

Given a 3-turn text conversation between two users, our system only relied on Turns 1 and 3 while discarding Turn 2 altogether. This was motivated by the insight that the label for each conversation was "the human judged label of *Emotion of Turn 3* based on the conversation" (from the competition website, emphasis ours). This means the overall emotional context of the conversation is dictated by the *reaction* of User A to the chatbot User B's response to A's initial post in Turn 1. Therefore, we hypothesised that the most informative features regarding the emotional context would come from User A's text (with Turn 3 likely to be more informative than Turn 1), and that Turn 2 would largely degrade the impact of these features. This was borne out by experimentation, where inclusion of features from Turn 2 only ever decreased the score compared with models using only data from Turn 1 and Turn 3. Table 2 shows the results of our experiments using different combinations of turns in terms of F1 score.

Turns 1 and 3 were then tokenized using the NLTK Tweet Tokenizer with additional rules[1] to handle symbols, and odd tokens.

These tokens were then scanned for negation words such as {`'not'`, `'isnt'`, `'no'`, `'dont'`} and combined with the next neighbouring token to create one new token. For example, the sentence: `'not happy jane'` tokenized turns into `'not'`, `'happy'`, `'jane'` finally combining negations turns into `'not_happy'`, `'jane'`.

Next the tokens were turn-encoded, meaning each token is prepended with `'1_'` or `'3_'` based on which turn the token was extracted from. For example, tokens from Turn 1 {`'you'`, `'are'`, `'funny'`} are prepended to {`'1_you'`, `'1_are'`, `'1_funny'`}, while Turn 3 tokens {`'=)'`} turn into {`'3_=)'`}.

3.2 Feature selection using TFIDF

The previous section constructs all the tokens found in the `Training`, `Dev` and `Test` data. We then use only the vocabulary found in the intersection of all the datasets. This was done because features which do not appear in the test set have no predictive power on that set. We found in practice that including these features did not improve the scores obtained by our system.

The term frequency inverse document frequency (TFIDF) score was then computed for each term per label basis. The token importance score is computed using the highest TFIDF score for that token across classes minus the second highest. In this manner the importance score used was a measure of a feature's ability to discriminate between classes. Experimentation on the `Dev` set showed that this produced better results than filtering based on, for example, the highest TFIDF score across classes. Figure 2 shows the top 10

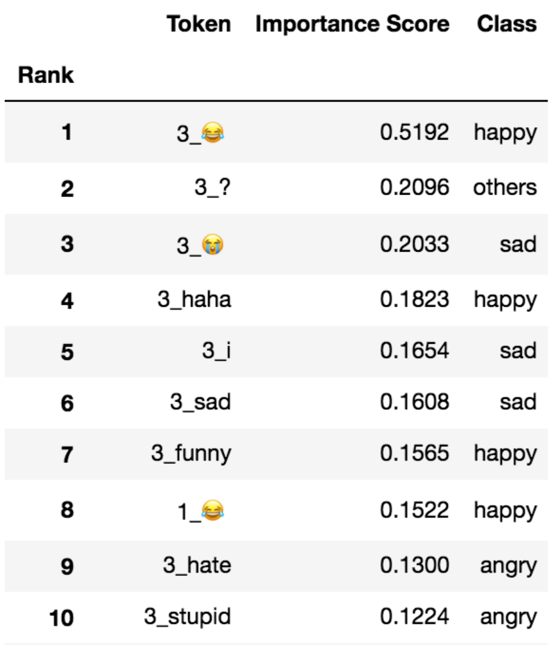
[1]https://github.com/andrew-ai/EmoContext2019/blob/master/tokenzie.py

features based on the importance scores.

A cutoff importance score was experimentally selected to further filter down the vocabulary to use only the most informative tokens and avoid overfitting (see Figure 3).

We note that the system is robust to variations in this parameter above a certain threshold – as shown in Figure 3, while the F1 score varies greatly for cutoff values between 0 and 0.005, above 0.005 the F1 score varies only between 0.720 and 0.725.

The features were then one-hot encoded as our final transformation.



Rank	Token	Importance Score	Class
1	3_😂	0.5192	happy
2	3_?	0.2096	others
3	3_😭	0.2033	sad
4	3_haha	0.1823	happy
5	3_i	0.1654	sad
6	3_sad	0.1608	sad
7	3_funny	0.1565	happy
8	1_😂	0.1522	happy
9	3_hate	0.1300	angry
10	3_stupid	0.1224	angry

Figure 2: Top 10 features based on the importance score computed and their associated labels.

3.3 Model pipeline: Core Model

The sklearn Linear Support Vector Classifier (LinearSVC) was used as our classifier. We used a gridsearch strategy with a simple hold out set (`Training` set to train and `Dev` set to validate) to tune the best performing hyper parameters based on F1 score for the `Training` and `Dev` datasets. We used sklearn's default LinearSVC parameters, except for the cost penalty (C) which we tuned and set to $C = 0.3$.

3.4 Model pipeline: Class tuning

Once the core model and model parameters were well tuned, we further tuned the predictions output by the system. Each class of predictions was

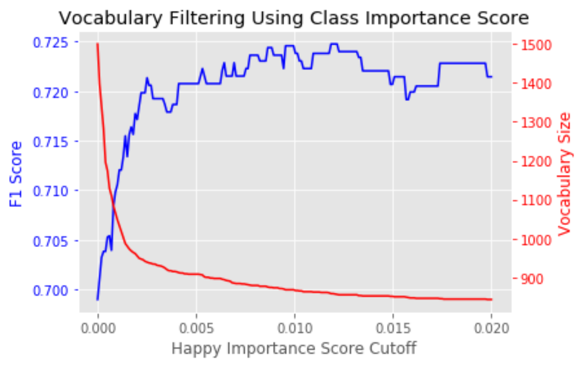


Figure 3: F1 score and vocabulary size as a function of the happy importance score cutoff. The importance scores for the `angry`, `sad` and `others` classes was set to 0.001 from prior tuning. Tuning individually on the `happy` importance score was performed due to it being the weakest classification class and in the end, 0.007 was chosen, which was a local max.

ranked and the least confident predictions based on sklearn's LinearSVC decision_function were shifted to the `others` class. Shifting the classes to have `angry` $\approx$ 5%, `happy` $\approx$ 4.5% and `sad` $\approx$ 4% maximised our F1 score. The results of these experiments are shown in Table 2. We note that this relies on the observation that the `Dev` and `Test` sets have approximately the same class-wise distributions, which of course may not hold true if the model is to be used in other contexts.

3.5 Model pipeline: Happy vs Rest override

This model tended to under-predict the `happy` class in terms of precision and recall, therefore an additional model was built to boost classifications within this class. A Happy vs Rest classifier was trained, using the same LinearSVC model and parameters but this time trained on the Training data relabeled as *'happy'* or *'rest'* (`angry`, `sad`, `others`).

After the Core Model predictions and the class tuning, the Happy vs Rest predictions were compared and the most confident predictions based on a confidence threshold was used to override the Core Models predictions. This threshold was chosen by ranking the predictions from the Happy vs Rest classifier in decreasing order of confidence, and converting the top n most confident predictions to `happy` over this threshold. In this case, all predictions from the Happy vs Rest linearSVC decision function with a signed distance over -0.6 were converted. Here the threshold was chosen by inspection of where the predictions appeared to change from being correct to dubious.

This allowed a more focused model to boost `happy` predictions that the Core Model had missed.

4 Results

Table 2 shows the results of all the system variants described in Section 3. The best overall micro F1 score achieved by our system was 0.735719, or 92.4% of the overall top score in the competition. This placed us in the top 22% of all submissions, and resulted in a final ranking of 37th place.

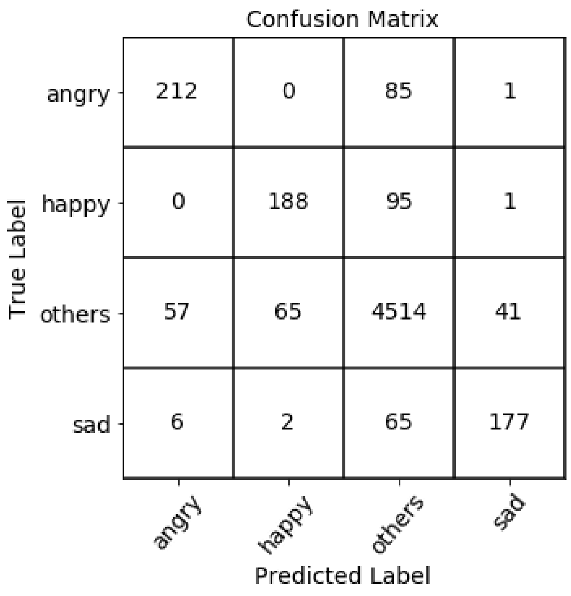
We note that for each variant of the model, using only Turn 1 + Turn 3 consistently improved the F1 score over using all turns. Adding the Happy vs Rest classifier boosted the `happy` predictions from: Precision = 0.7490, Recall = 0.6514, F1 = 0.6968 to Precision = 0.7368, Recall = 0.6901, F1 = 0.7127. This boosted our overall micro F1 score from 0.729352 to 0.735719. While not shown in Table 2, our *post hoc* class balancing of the final predictions to match the proportions in the `Dev` set was consistently also beneficial in improving our model, as shown in Figure 3.

	T123	T13
NCB	0.7155	0.7188
CB	0.7255	0.7294
NCB + HvR	0.7175	0.7198
CB + HvR	0.7294	**0.7357**

Table 2: Experimental results for system variants where: *T123* = Turn 1 + Turn 2 + Turn 3, *T13* = Turn 1 + Turn 3, *NCB* = No class balancing, CB = Class balancing against test set, *HvR* = With Happy vs Rest override. Note that in each case using using T13 improves over using T123.

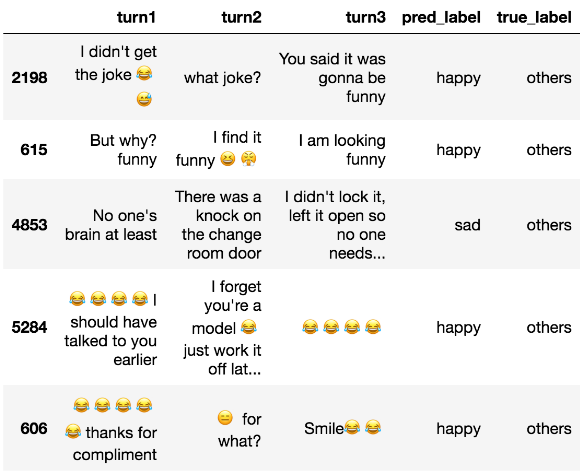
Figure 4 shows a confusion matrix for predictions made by our system. In general, the worst misclassifications made by our system were predictions of the `others` class when the correct label was one of the other emotions (or vice versa), rather than misclassifications between emotions, which were negligible.

Figure 5 shows the top 5 "mistakes" – where our system was highly confident in its prediction, but turned out to be wrong. All of these top mislabels were when we incorrectly predicted an emotional class (`angry`, `happy`, `sad`) rather than `others`, and generally when the text contained emojis.



	angry	happy	others	sad
angry	212	0	85	1
happy	0	188	95	1
others	57	65	4514	41
sad	6	2	65	177

Figure 4: Confusion matrix of misclassifications by our system.



	turn1	turn2	turn3	pred_label	true_label
2198	I didn't get the joke 😂 😅	what joke?	You said it was gonna be funny	happy	others
615	But why? funny	I find it funny 😆 😤	I am looking funny	happy	others
4853	No one's brain at least	There was a knock on the change room door	I didn't lock it, left it open so no one needs...	sad	others
5284	😂 😂 😂 😂 I should have talked to you earlier	I forget you're a model 😂 just work it off lat...	😂 😂 😂 😂	happy	others
606	😂 😂 😂 😂 😂 thanks for compliment	😑 for what?	Smile😂 😂	happy	others

Figure 5: Top misclassifications by our system. All errors were when the correct label was "others" but our system predicted an emotion.

5 Conclusion

That our system performed so well using such a simple approach was very satisfying, and demonstrates that high-quality emotion detection is achievable by practitioners across a range of disciplines. Future work using this system will investigate a longitudinal study of long-term changes in the emotional context of conversations conducted over online social network platforms.

References

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Emily M Cody, Andrew J Reagan, Lewis Mitchell, Peter Sheridan Dodds, and Christopher M Danforth. 2015. Climate change Sentiment on Twitter: An unsolicited public opinion poll. *PLoS ONE*, 10(8):e0136092.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A Sentiment-and-Semantics-Based Approach for Emotion Detection in Textual Conversations. In *Proceedings ofNeu-IR 2017 SIGIR Workshop on Neural Information Retrieval, Shinjuku, Tokyo, Japan, August 11, 2017 (Neu-IR '17)*, pages 1–6.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

Adam S Miner, Arnold Milstein, Stephen Schueller, Roshini Hegde, Christina Mangurian, and Eleni Linos. 2016. Smartphone-based conversational agents and responses to questions about mental health, interpersonal violence, and physical health. *JAMA Internal Medicine*, 176(5):619–625.

Jonathan Tuke, Andrew Nguyen, Mehwish Nasim, Drew Mellor, Asanga Wickramasinghe, Nigel Bean, and Lewis Mitchell. 2018. Pachinko Prediction: A Bayesian method for event prediction from social media data. *arXiv preprint: 1809.08427*.

SCIA at SemEval-2019 Task 3: Sentiment analysis in textual conversations using Deep Learning

Zinedine Rebiai [1]
zinedine.rebiai@epita.fr

Simon Andersen [1]
simon.andersen@epita.fr

Antoine Debrenne [1]
antoine.debrenne@epita.fr

Victor Lafargue [1]
victor.lafargue@epita.fr

[1] EPITA Graduate School of Computer Science, France

Abstract

In this paper we present our submission for SemEval-2019 Task 3: EmoContext. The task consisted of classifying a textual dialogue into one of four emotion classes: happy, sad, angry or others. Our approach tried to improve on multiple aspects, preprocessing with an emphasis on spell-checking and ensembling with four different models: Bi-directional contextual LSTM (**BC-LSTM**), categorical Bi-LSTM (**CAT-LSTM**), binary convolutional Bi-LSTM (**BIN-LSTM**) and Gated Recurrent Unit (**GRU**). On the leader-board, we submitted two systems that obtained a micro F1 score ($F1\mu$) of **0.711** and **0.712**. After the competition, we merged our two systems with ensembling, which achieved a $F1\mu$ of **0.7324** on the test dataset.

1 Introduction

Rapid progress in natural language processing with the rise of deep learning has brought increasing attention on tasks such as text classification and sentiment analysis. Most of the work in that field was made using social media due to the large amount of data available. The task addressed in this paper focuses on emotion detection within conversations from social media. The key point is that we need to take into account multiple speakers and capture a global emotion out of their conversation. It becomes a challenge when facing different users who each have a different way to express their emotions depending on their personalities. In a dialogue, users have an initial emotional state, and their mood will shift as the dialogue goes on. Therefore, the task of labelling a turn-based conversation with the right emotion is even more challenging.

State of the art approaches consist of using language models (Vaswani et al., 2017) to pre-train the model on the general NLP task of language modeling before fine-tuning on specific tasks like classification or translation. The language model approach used by **ULMFiT** (Howard and Ruder, 2018), **ELMO** (Peters et al., 2018) and **BERT** (Devlin et al., 2018) was especially successful for this kind of tasks. For the specific task of emotion classification in textual conversation, Gupta et al. (2017) achieved a $F1\mu$ score of 0.7134 on the same dataset, using an architecture based on **LSTM**. For sentiment analysis, other successful approaches also used **Bi-LSTM** (Baziotis et al., 2017b) as well as transfer learning (Daval-Frerot et al., 2018).

In this paper, we present two sub-systems that are composed of four deep-learning models (using **Bi-LSTM**, **GRU** and **CNN**). Those two sub-systems competed independently at SemEval-2019 Task 3 (Chatterjee et al., 2019). After the final evaluation, we merged both sub-systems, taking advantage of ensemble learning. The rest of the paper is organized as follows. Part 2 gives an overview of our approach. Our preprocessing methods, the description of our models, and our ensembling approach are all described in Part 3. Part 4 shows the obtained results and in Part 5, we give a conclusion with remarks for future works.

2 Overview

Our four models are: **CAT-LSTM**, **BIN-LSTM**, **BC-LSTM** and **GRU**. We decided to use four different model architecture, two different preprocessing and two different word embeddings. We built very diverse models in order to maximize the effect of ensembling on our system. **CAT-LSTM** and **BIN-LSTM** share the exact same preprocessing and embeddings, while **BC-LSTM** and **GRU** use the same embeddings but slightly different preprocessing methods.

3 Proposed System

3.1 Text Preprocessing

We used two different preprocessing methods. However, both preprocessing share the same normalization (all words are lower cases and numbers, links, emails and dates were replaced by special tags). String emoticons are transformed into Emojis before tokenization (':)' becomes 😃).

3.1.1 CAT-LSTM and BIN-LSTM

Here, the preprocessing used was motivated by the fact that the dataset comes from social media, meaning the writing style contains improper use of grammar, misspellings, emoticons and slang. Because of that we used the *ekphrasis*[1] (Baziotis et al., 2017a) library which was made specifically for preprocessing text from social networks. This tool performs tokenization, word normalization, word segmentation and spell correction. Here, we didn't take into account the fact that our inputs are turned based and instead we added a special token `<eos>` in-between each turn of the dialogues which we then concatenated together. We also improved **spellchecking** with this method. Indeed, we realized that 8.8% of our vocabulary consisted of words that weren't part of our word embeddings because they were misspelled, even after preprocessing with *ekphrasis*. Obvious spelling errors like *angru* instead of *angry* were still present. In order to solve this problem we used a spellchecking library named *autocorrect*[2] after preprocessing with *ekphrasis* which decreased to 3.4% the number of unknown words from our vocabulary.

3.1.2 BC-LSTM and GRU

Here, for the normalization, specials tags (numbers, links, emails and dates) were removed. We did the **spellchecking** ourselves with the most common mistakes (e.g: *waht* becomes *what*). This makes the conversation cleaner and easier to understand while leaving a part of natural since most of the words are not corrected (e.g: *nooo* stay *nooo* instead of just *no* since it has a stronger meaning). Notice that for our **GRU** model, we concatenated the three input turns to make it simpler for the **GRU** to process but for our **BC-LSTM**, we made separate layers to process each turns.

[1] https://github.com/cbaziotis/ekphrasis

[2] https://github.com/phatpiglet/autocorrect

3.2 Pre-Trained Word Embeddings

Word embeddings are dense vectors representing semantic meaning for each word of the vocabulary. We used pre-trained word embeddings to initialize the weights of our embedding layers. **CAT-LSTM** and **BIN-LSTM** used *Datastories* embeddings [3], while **BC-LSTM** and **GRU** used our own pre-trained word embeddings.

3.2.1 CAT-LSTM and BIN-LSTM

The weights we used for the embedding matrix of these models were the same as Baziotis et al. (2017b), pre-trained on 330 millions of english tweets messages posted from 12/2012 to 07/2016 with GloVe.

3.2.2 BC-LSTM and GRU

For these models, each word is represented by a vector of 312 dimensions which are obtained by the concatenation of the following features :

- *Word2Vec* (Mikolov et al., 2013) - We created our vector representations of words using *Word2Vec* networks trained with skip-gram and negative sampling on 30 millions of english tweets messages posted from 01/2017 to 06/2017. This word embeddings are 300 dimensional.

- *Affect Intensity Lexicons* (Mohammad and Turney, 2013) - 6,000 entries for four basic emotions: anger, fear, joy, and sadness. Considering fear as other, it adds 4 dimensions

- *Emolex* (Novak et al., 2015) - The NRC Emolex is a list of words and their associations with eight emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust). We transformed fear, anticipation, trust, surprise and disgust into the class 'others' with a vector of 4 dimensions (joy, anger, sad, others)

- *Emoji Emotion* [4] - List of emoji rated by polarity. The polarity was hand classified (by one person) based on the names of these emoji. The contained emoji are the faces defined by Unicode

[3] https://github.com/cbaziotis/datastories-semeval2017-task4

[4] https://github.com/words/emoji-emotion

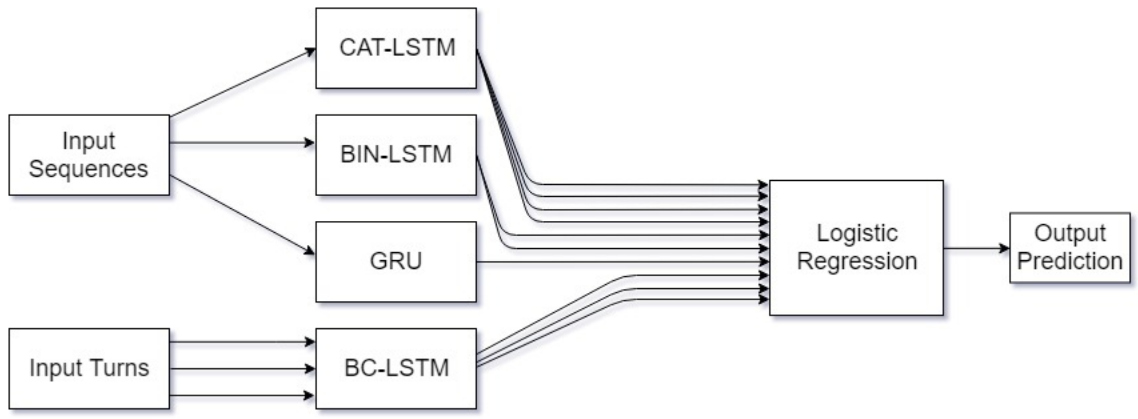


Figure 1: Ensembling method.

3.3 Models Description

We used Bi-directional Long Short-Term Memory networks (*B-LSTM*) in every model except the **GRU**. Every model used *Adam* optimizer and *crossentropy* [5] as the loss function.

3.3.1 CAT-LSTM Model

The core of the network is composed of two sets of *B-LSTM* as in Baziotis et al. (2017a). The input layer is composed of an embedding layer of size 300, followed by a dropout layer (0.4) directly after the embedding layer to help regularizing by showing slightly different sequences every epochs. Each *B-LSTM* layer consist of 150 units with recurrent dropout (0.5) and regular dropout (0.5). *B-LSTM* layers reads each sequence two times in different order, forward (from left to right) and backward (from right to left) which helps to capture the context of the sentence. The output layer is a dense layer followed by a softmax.

3.3.2 BIN-LSTM Model

With the previous model (**CAT-LSTM**), we realized most of our errors came from confusion between the class ’others’ and the rest (angry, happy and sad). Which might be because the training set is slightly unbalanced (15k others, 5.5k angry, 5.4k happy, 4.2k sad). Because of that we decided to train a model specifically on binary classification between the class ’others’ and the rest. That way we could use ensembling to help our categorical model to differentiate between the two categories. For this binary model, we kept a similar architecture as **CAT-LSTM**. However, we added a convolution 1D and a maxpool before the first *B-LSTM* layer to train faster. Since our input is a one-dimensional sequence of words, it makes sense to use a one-dimensional convolution right after the embedding layer in order to capture meaningful context about our sequence while reducing its size. We kept most spatial information by using a kernel size of 5 so that we take every group of 5 adjacent words into account. After the convolution, we used a one-dimensional maxpool layer of size 5 in order to reduce the input size. With this new architecture we were able to train a second model more focused on separating ’others’ class from the rest of the emotion classes while increasing our training speed by 80%.

[5]We used *categorical crossentropy* except for the binary (**BIN-LSTM**) model which used *binary crossentropy*

3.3.3 BC-LSTM Model

We used the **BC-LSTM** architecture introduced in Poria et al. (2017). **BC-LSTM** (Bidirectional Contextual LSTM) is a model for context-dependent sentiment analysis and emotion recognition. For this model we treated each turn inputs in separate parallel layers before concatenating the results. Hence, we have 3 parallel embedding layers with our pre-trained embeddings, which then go through 3 parallel bidirectional LSTM layers of 300 units. Each of those *B-LSTM* have a Dropout of 0.5. After that, we stack each *B-LSTM* layer. We then have a fully-connected layer with 4 units, that will give us probabilities for each emotion class. This model was particularly useful to detect happy, sad, and others emotions so we only keep those 3 probabilities.

3.3.4 GRU Model

The **GRU** model (Cho et al., 2014) allowed us to discriminate the angry class. The first layer is made using our pre-trained embeddings to process the concatenated text input. We added a dropout of 0.5. Then, the output of our embedding goes through a *GRU* layer of 128 units. On top of our *GRU*, we added a fully-connected layer of 32 units with relu as the activation function. A Dropout of 0.2 was added after this layer.

3.4 Ensembling

As stated before, we trained each model individually, hence giving us multiple sets of predictions for each input sample. We used all of those probabilities to train a logistic regression. We stack all 10 predictions (4 from **CAT-LSTM**, 2 from **BIN-LSTM**, 1 from **GRU** and 3 from **BC-LSTM**) to a create a new training sample associated with the corresponding true label of the sample, as shown in Figure 1. This way, we take each of our models into account and the logistic regression takes care of weighting the importance of our models. Since our four models are very diverse, they all contribute to the final prediction.

4 Results and Analysis

When evaluating each group of model separately, we found that they were correct on different samples even if the $F1\mu$ score is almost the same. Table 1 illustrate the performances of our systems on the test set. We can see that the class 'happy' gave our models the most trouble. Which might be because it is the smallest class in the dataset. Ensembling had a little impact on this emotion compared to 'angry' and 'sad'. Our final system achieves an $F1\mu$ score of **0.7324**.

Emotion / Models	Angry	Happy	Sad	$F1\mu$
CAT-LSTM+ BIN-LSTM	0.719	0.678	0.734	0.711
GRU+ BC-LSTM	0.722	0.673	0.739	0.712
Model ensembling	0.744	0.689	0.766	**0.7324**

Table 1: $F1\mu$ score on the test set for each model.

Note that [**CAT-LSTM + BIN-LSTM**] and [**GRU + BC-LSTM**] were submitted independently for the final submission. However, both systems were combined in **model ensembling** after the competition ended which significantly improved our final score.

5 Conclusion

In this paper, we proposed to use ensemble learning for sentiment analysis in conversations (SemEval2019 Task 3). Using various neural networks structures such as **B-LSTM**, parallel **B-LSTM**, **GRU** and **CNN**, ensemble learning takes advantage of this diversity of approach to make a prediction for our emotions classes (angry, happy, sad or others). Each model was trained separately on the given corpus. Then, we trained a logistic regression with the probabilities given by our four deep learning models in order to make the final predictions for each conversations. Our ensembling system achieved a $F1\mu$ score of **0.7324** on the final testing set, after the competition ended.

Improvements could be made by gathering more models. A properly fine-tuned language model (for instance using ULMFiT) or LSTM with attention mechanism could improve our current system. Future work will consist of finding better ensembling methods and working with language models pre-trained on larger corpus of data.

Acknowledgments

We would like to thank Abdessalam Bouchekif and Anatole Moreau for interesting scientific discussions (Daval-Frerot et al., 2018).

References

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017a. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada. Association for Computational Linguistics.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017b. Datastories at semeval-2017 task 6: Siamese lstm with attention for humorous text comparison. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 381–386, Vancouver, Canada. Association for Computational Linguistics.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Kyunghyun Cho, Bart van Merrienboer, aglar Gülehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

Guillaume Daval-Frerot, Abdesselam Bouchekif, and Anatole Moreau. 2018. Epita at semeval-2018 task 1: Sentiment analysis using transfer learning approach. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 151–155. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv e-prints*, page arXiv:1810.04805.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations. *CoRR*, abs/1707.06996.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. *arXiv e-prints*, page arXiv:1801.06146.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Saif Mohammad and Peter D. Turney. 2013. Crowdsourcing a word-emotion association lexicon. *Computational Intelligence*, 29:436–465.

Petra Kralj Novak, Jasmina Smailovic, Borut Sluban, and Igor Mozetic. 2015. Sentiment of emojis. In *PloS one*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh, and Louis-Philippe Morency. 2017. Context-dependent sentiment analysis in user-generated videos. In *ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv e-prints*, page arXiv:1706.03762.

Sentim at SemEval-2019 Task 3: Convolutional Neural Networks For Sentiment in Conversations

Jacob Anderson
Sentim LLC, USA
papers@sentimllc.com

Abstract

In this work convolutional neural networks were used in order to determine the sentiment in a conversational setting. This paper's contributions include a method for handling any sized input and a method for breaking down the conversation into separate parts for easier processing. Finally, clustering was shown to improve results and that such a model for handling sentiment in conversations is both fast and accurate.

1 Introduction

The model for this paper was created for Task 3 of SemEval 2019, EmoContext (Chatterjee et al., 2019). The basic idea of the task is to classify the emotion (as "angry", "sad", "happy", or "others") that someone is expressing in the third turn of a three part conversation, given the previous two turns as context.

The dominant approach in many natural language tasks is to use recurrent neural networks or convolutional neural networks (CNN) (Conneau et al., 2016). For classification tasks, recurrent neural networks have a natural advantage because of their ability to take in any size input and output a fixed size output. This ability allows for greater generalization as no data is removed nor added in order for the inputs to match in length. While CNN's can also support input of any size, they lack the ability to generate a fixed size output from any sized input. In text classification tasks, this often means that the input is fixed in size in order for the output to also have a fixed size.

This work expands upon a previous work (Anderson, 2018), a way of using CNN's for classification to allow for any sized input length without adding or removing data. That work was expanded upon in this paper by making it simpler, using it in a different setting, and applying a new method to compensate for the previous model's deficiencies.

2 Model Description

The overall architecture of this model can be broken into two parts: the language understanding model and the emotion prediction model. The language understanding model takes in each part of the conversation and processes it separately in order to produce a latent vector representing the network's understanding of that part of the conversation. Then, the emotion prediction model takes all of the latent vectors from the language understanding model and combines them in order to make a prediction for what the emotion is.

More specifically, the language understanding model takes in the first and third turns of each conversation[1], processes them separately, and returns a respective understanding vector of size 128 for each input. The emotion model then concatenates those vectors to make one vector of size 256 and then processes that output through a small fully connected network in order to predict the emotional content.

The model was designed using Tensorflow (Abadi et al., 2015) and Keras (Chollet et al., 2015) and was trained using a Google Colab GPU.

2.1 Language Understanding Model

The idea behind the language understanding model (Figure 1) is to take in each part of the conversation separately and process them into a fixed size vector representing a machine understanding of the input text. In order to accomplish this, the input conversation was first embedded into a subword embedding using the byte-paired encoding provided by Heinzerling and Strube (2018) and SentencePiece (Kudo and Richardson, 2018). The subword embedding used a base vocabulary size of 10,000 with a vector size of 100. This em-

[1]The second turn was not used during my training because it was found empirically to make the results worse.

bedding was further extended with random vectors during data preprocessing to include emojis and other unique characters not found in the original vocabulary.

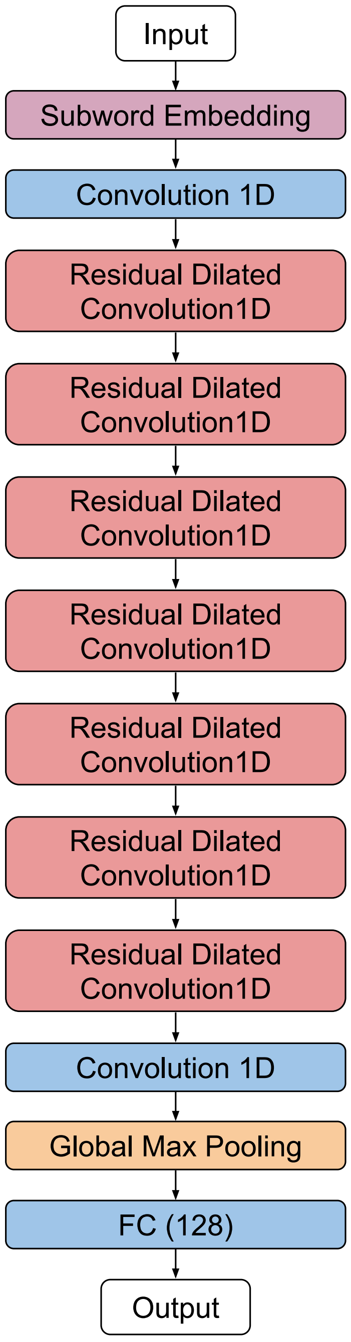

Input
Subword Embedding
Convolution 1D
Residual Dilated Convolution1D
Residual Dilated Convolution1D
Residual Dilated Convolution1D
Residual Dilated Convolution1D
Residual Dilated Convolution1D
Residual Dilated Convolution1D
Residual Dilated Convolution1D
Convolution 1D
Global Max Pooling
FC (128)
Output

Figure 1: The language understanding model.

A specialized convolutional stack was used in order to process the embeddings. The first layer of the stack was a convolutional layer with linear activation in order to set up the initial size of the input. This was followed by a stack of seven residual dilated convolutional layers (Figure 2) with relu activation. The number of layers was chosen so that the final layer would have a receptive field large enough to be able to see the whole input.

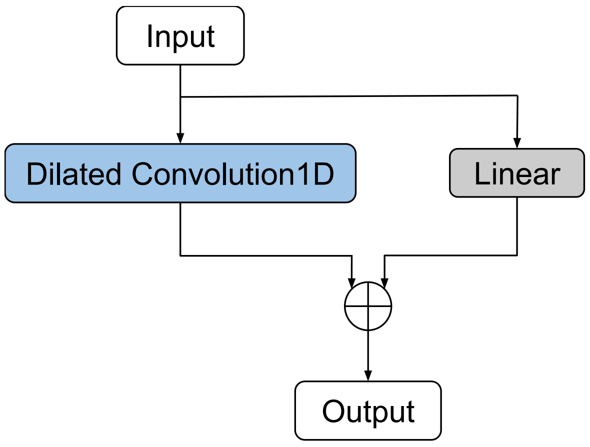


Figure 2: A residual dilated convolution, without skip connections.

One more convolutional layer was used with a linear activation function and the “same” padding method. This layer was originally added to increase the number of filters, but after a grid search was performed to determine the optimal hyperparameters, this layer was left in as a way to do any necessary linear transformations on the latent vector. Note that all convolutional layers in the final version had a kernel size of 2 and 300 filters. Additionally, the padding method varied depending on the experiment. See the “Experiments and Results” section for more information on that topic.

The convolutional stack was followed by a global max pool layer to bring the network down to a constant size, and then an FC Layer. The idea of this last FC layer is to allow for any fine tuning of the language understanding model as necessary and to compensate for the max pool layer’s inability to provide perfect information.

2.2 Emotion Prediction Model

The purpose of the emotion prediction model (Figure 3) is to predict the emotion given the latent vectors representing the conversation. The emotion prediction model starts off with the concatenation all of the language understanding outputs. This is then fed through one fully connected layer of size 256 with sigmoid activation, followed by the final prediction fully connected layer of size 4, also with sigmoid activation.

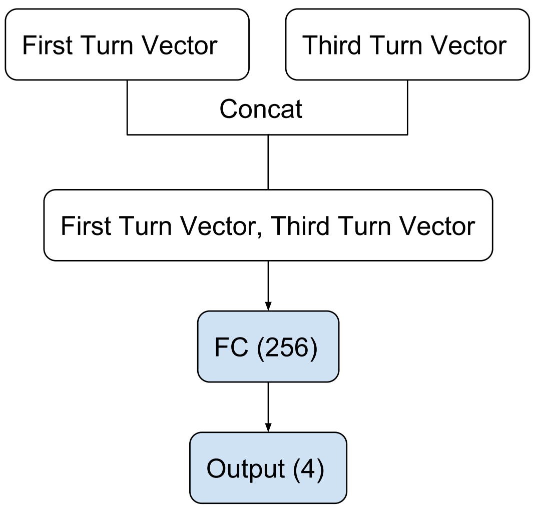


Figure 3: The layout of the emotion prediction model.

2.3 Training and Loss Functions

The network is end to end trainable without having to train the networks separately. Triplet loss was used (with triplets chosen via the all anchor-positive method) from Schroff et al. (2015) to cluster similarly labeled data together for the language understanding model, and softmax cross entropy loss was used for the emotion prediction model. The models were trained using the Adam (Kingma and Ba, 2014) optimization method with a learning rate of .001.

The loss function can be written as

$$L = T(a_s, p_s, n_s) + T(a_e, p_e, n_e) + S(labels, logits)$$

Where $T(anchor, positive, negative)$ is the triplet loss and $S(labels, logits)$ is the softmax cross entropy loss given the true labels and logit predictions from the emotion prediction model. Additionally, a_s, p_s, n_s are the anchor, positive, and negative examples corresponding to the first turn and a_e, p_e, n_e are the anchor, positive, and negative examples corresponding to the last turn respectively.

To further clarify, each first turn text in the batch was matched against every other first turn text so that they (the anchor) would be closer to similarly labeled data (positive examples) and farther away from every other category of data (the negative examples). The same thing was done for the last turn of the conversation. The first turn was specifically not clustered to be close to both first turn and last turn labels in order to preserve any contextual information that could be available in the first turn of a conversation versus the last turn.

An interesting thing to note is the time taken to train the model. Perhaps due to using the clustering loss and the softmax cross entropy loss simultaneously, the fast speed of training residual convolutions, or the small size of the dataset, the model always finished training within 7 minutes (or 9 epochs). Any more than that would never improve the micro f1 score and would start overfitting. Furthermore, roughly twenty percent of the time the model would show the best results within 1.5 minutes (or 2 epochs).

In one experiment, the networks were trained using just softmax cross entropy loss, but the networks never improved beyond the initial prediction of always pick the “others” class. This could be because of wrong hyperparameters (the same model was used as the one with the clustering loss), didn’t train for long enough (one model was trained for 24 hours (or 1755 epochs over the training data)), or didn’t try enough random restarts (10 restarts were attempted in this paper). Regardless of the case, this shows that using the clustering loss helped the network find the right solutions faster and with less hyperparameter tuning than just using the softmax cross entropy loss.

3 Experiments and Results

The experiments for this paper were all run on the dataset provided by the EmoContext organizers. The main dataset contains roughly 30,000 conversations, while the test dataset contains roughly 2,750, and the final evaluation dataset contains about 5,500. The order of each conversation goes Person A, Person B, then Person A again for all of the datasets.

Model Type	**Micro F1 Score**
118 network ensemble	.7295
Same padding ensemble	.7262
Causal padding ensemble	.7357
Best single model	.7255
2nd best single model	.707
Ensemble of micro f1 $>.7$	.7366
Ensemble of micro f1 $>.71$	.7386
Emoji replacement model	.7386

Table 1: Micro f1 score on the final evaluation set for each of my submissions.

The first submission on the evaluation dataset was an ensemble of 118 networks. In total, the 118 networks came from 100 different training cycles: 50 runs using causal padding as the padding method in the dilated residual convolutional stack, and 50 runs using the "same" padding method. Each run was trained against the full dataset for nine epochs and tested against the test dataset, so the checkpoints chosen were the ones that performed the best on the test dataset. The remaining 18 networks came from checkpoints that were not the best checkpoint of the training run but still had a micro f1 score above 0.7 on the test set.

To compare two different convolutional padding methods, an ensemble of all networks using "same" padding (56 in total), and an ensemble of all causal padding runs (62 in total) was submitted. While the causal padding models performed better on average, the difference was not significant. Two non-ensemble runs were submitted - one of which was the best run of all 100 runs (which happened to be a causal padding model) and the other of which was the second best run of all 100 runs (which happened to be a "same" padding model).

The sixth submission was an ensemble where every network in the ensemble had to have a micro f1 score of .7 or higher and the seventh submission was an ensemble where every network had to have a micro f1 score of .71 or higher respectively.

Unsurprisingly, the ensemble methods ourperform the best single model runs on micro f1 score. Interestingly though, the 118 network ensemble performed worse than the causal padding ensemble. This could represent opposing learning biases present in "casual" versus "same" padding types. That is, instead of the two different types of models agreeing with each other and removing bad predictions, the two models instead mostly disagree over certain portions of the data, leading to larger uncertainty and therefore lower overall performance.

A rule based model was also tested where every conversation was labeled based on whether or not it contained an emoji. More specifically, for every conversation the first and third turns of the text were concatenated and then if that had a happy emoji it was labeled happy, if it had a sad emoji it was labeled sad, if it had an angry emoji it was labeled angry, and if it didn't have any of those emojis then it was labeled using the output of the seventh submission. This model is interesting because applying the algorithm changed zero labels from the seventh submission. This means that the network learned the different emojis and used them in order to predict the emotion. This also agrees with the intuition that emojis should represent how a user is feeling (or perhaps that the dataset was created or biased by this idea). To see all of the results, view Table 1.

Submitter	**F1 Score**
1st Place	.7959
2nd Place	.7947
3rd Place	.7765
My Best	**.7386**
Median	.6947
Average	.6605

Table 2: Micro f1 score of the first, second, and third place submissions as well as the average and median scores of all submissions as compared to my best submission.

To see a comparison to other people's submissions in the competition, view Table 2. Additionally, I show the micro f1 score as well as the f1 score for the happy, angry, and sad labels for a few of my models in Table 3.

Model	**F1**	**Angry**	**Happy**	**Sad**
BSM	.7255	.7382	.6718	.7653
f1 >.7	.7366	.7481	.6812	.7803
f1 >.71	.7386	.7536	.6790	.7818

Table 3: Micro f1 score as well as the f1 score for the angry, happy, and sad labels. Best single model is abbreviated as BSM, as well as the ensemble of all models that scored higher than a certain f1 score as f1 >score.

4 Conclusion

A language understanding model and an emotion prediction model was trained in an end to end fashion for understanding and predicting the emotions of conversations. The previous work was adapted in order to leverage the ability to use any sized inputs with convolutional neural networks, and showed that such a model can be fast and accurate. Finally, it was shown that augmenting language learning using clustering loss can help augment training and improve results.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Jacob Anderson. 2018. Fully convolutional networks for text classification. *arXiv preprint arXiv:1902.05575*.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

François Chollet et al. 2015. Keras.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*.

Benjamin Heinzerling and Michael Strube. 2018. BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *CoRR*, abs/1808.06226.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832.

SINAI at SemEval-2019 Task 3: Using affective features for emotion classification in textual conversations

Flor Miriam Plaza-del-Arco, M. Dolores Molina-González,
M. Teresa Martín-Valdivia, L. Alfonso Ureña-López

Department of Computer Science, Advanced Studies Center in ICT (CEATIC)
Universidad de Jaén, Campus Las Lagunillas, 23071, Jaén, Spain
{fmplaza, mdmolina, maite, laurena}@ujaen.es

Abstract

Detecting emotions in textual conversation is a challenging problem in absence of nonverbal cues typically associated with emotion, like facial expression or voice modulations. However, more and more users are using message platforms such as WhatsApp or telegram. For this reason, it is important to develop systems capable of understanding human emotions in textual conversations. In this paper, we carried out different systems to analyze the emotions of textual dialogue from SemEval-2019 Task 3: EmoContext for English language. Our main contribution is the integration of emotional and sentimental features in the classification using the SVM algorithm.

1 Introduction

Emotions seem to govern our daily lives since most of our decisions are guided by our mood. They are complex and that is why they have been studied in many areas over time. Given the importance to develop systems to be able to mimic functioning of the human brain, emotions have attracted the attention in the field of affective computing (Thilmany, 2007).

To our knowledge, there are not many works that focus on studying how emotions are reflected verbally. However, studying emotions on text messaging platforms such as WhatsApp, Facebook Messenger or Telegram is important as more and more users are using them to share their experiences and emotions.

Currently, detecting emotions in instant messaging has multiple applications in different fields (Gupta et al., 2017; Yadollahi et al., 2017; Hakak et al., 2017), such as businesses intelligence to increase customer satisfaction knowing their preferences, social media to alert users if they are going to post an offensive tweet or psychology to detect some disorders like anorexia, anxiety or stress.

In this paper, we present the different systems we developed as part of our participation in SemEval-2019 Task 3: Contextual Emotion Detection in Text (EmoContext) (Chatterjee et al., 2019b). It is an emotion classification task. Given a textual dialogue along with two turns of context, its consists of classify the emotion of user utterance as one of the emotion classes: Happy, Sad, Angry or Others.

The rest of the paper is structured as follows. In Section 2 we explain the data used in our methods. Section 3 introduces the lexical resources used for this work. Section 4 presents the details of the proposed systems. In Section 5, we discuss the analysis and evaluation results for our system. We conclude in Section 6 with remarks and future work.

2 Data

To run our experiments, we used the English datasets provided by the organizers in SemEval19 Task 3 : EmoContext (Chatterjee et al., 2019b). The datasets containing 3-turn conversations along with their emotion class labels (Happy, Sad, Angry, Others) provided by human judges. The Turn 1 contains the first turn in the three turn conversation, written by User 1. The turn 2 contains the second turn, which is a reply to the first turn in conversation and it is written by User 2 and finally, the turn 3 contains the last turn, which is a reply to the second turn in the conversation, which is written by User 1.

During pre-evaluation period, we trained our models on the train set, and evaluated our different approaches on the dev set. During evaluation period, we trained our models on the train and dev sets, and tested the model on the test set. Table 1 shows the number of 3-turn conversations used in our experiments in English.

Dataset	train	dev	test
Happy	4,243	142	284
Sad	5,463	125	250
Angry	5,506	150	298
Others	14,948	2,338	4,677
Total	30,160	2,755	5,509

Table 1: Number of 3-turn conversations per EmoContext dataset

3 Resources

For the development of the task, we used different lexicons that we explain in detail below.

NRC Affect Intensity Lexicon (Mohammad, 2017). It has almost 6,000 entries in English. Each of them has an intensity score associated to one of the following basic emotions: anger, fear, sadness and joy. The scores range from 0 to 1, where 1 indicates that the word has a high association to the emotion and 0 that the word has a low association to the emotion.

NRC Word-Emotion Association Lexicon (EmoLex) (Mohammad and Turney, 2010). This lexicon has a list of English words associated to one or more of the following emotions: anger, fear, anticipation, trust, surprise, sadness and joy.

VADER (Valence Aware Dictionary and sEntiment Reasoner) (Gilbert, 2014). The VADER sentiment lexicon is a rule-based sentiment analysis tool. This is sensitive both the polarity and the intensity of sentiments expressed in social media contexts, and is also generally applicable to sentiment analysis in other domains. VADER has been validated by multiple independent human judges. The tool returns four values: positive, negative, neutral and compound. The first three scores represent the proportion of text that falls in these categories. The compound score is computed by summing the valence scores of each word in the lexicon, adjusted according to the rules, and then normalized to be between -1 (most extreme negative) and +1 (most extreme positive).

4 System Description

In this section, we describe the systems developed for the EmoContext task. During our experiments, the scikit-learn machine learning in Python library (Pedregosa et al., 2011) was used for benchmarking.

4.1 Data Preprocessing

In first place, we preprocessed the corpus of conversations provided by the organizers. We applied the following preprocessing steps: the documents were tokenized using NLTK Tweet Tokenizer[1] and all letters were converted to lower-case.

4.2 Feature Extractor

Converting sentences into feature vectors is a focal task of supervised learning based sentiment analysis method. Therefore, the features we chose in our system can be divided into three parts: statistic features, morphological features and lexical features.

- **Statistic features**. We employed the feature that usually perform well in text classification: Term Frecuency (TF) taking into account unigrams and bigrams.

- **Morphological features**. We employed Part-of-speech tagging (PoS). For each sentence, we obtain a vector associated with the part of speech recognized in each word of the sentence.

- **Lexical features**. As we explained in Section 3, we used three lexicons obtained different features in the following way:

  1. **NRC Affect Intensity**. We checked the presence of lexicon terms in the sentence and then we computed the sum of the intensity value of the words of the sentence grouping them by the emotional category (*fear*, *sadness*, *anger* and *joy*). Therefore, we obtained a vector of four values (four emotions) for each sentence. Each value of intensity is normalized $\hat{i}_e$ applying the following equation:

  $$\hat{i}_e = \frac{i_e}{\sum_e i_e}$$

  Where $e = \{\text{fear, sadness, anger, joy}\}$ and i_e is equal to value of intensity per emotion. Note that the components of the normalized vector add up to 1, and each of them is a positive number between 0 and 1.

[1]https://www.nltk.org/api/nltk.tokenize.html

2. **Emolex**. We identified the presence of lexicon terms in the sentence and we assigned 1 as confidence value (CV). Then, we summed the CV of the words whose emotion is the same obtaining a vector of emotions for each sentence. As a result, we obtained a vector of eight values (eight emotions). Each value $\hat{v}_e$ is normalized following the next equation:

$$\hat{v}_e = \frac{CV_e}{\sum_e CV_e}$$

Where e = {anger, fear, anticipation, trust, surprise, sadness and joy} and CV_e is equal to confidence value per emotion. Note that the components of the normalized vector add up to 1, and each of them is a positive number between 0 and 1.

3. **VaderSentiment**. We use the sentiment.vader module[2] provided by the Natural Language Toolkit (NLTK). With this module, we analyze each sentence and we obtained a vector of four scores: negative sentiment, positive sentiment, neutral sentiment and compound polarity.

4.3 Classifier

The concatenation of the features described before are applied for the classification using the SVM algorithm. We selected the Linear SVM formulation, known as C-SVC and the value of the C parameter was 1.0.

5 Experiments and analysis of results

During the pre-evaluation phase we carried out several experiments and the best experiments were taken into account for the evaluation phase. The architecture of the different systems can be seen in Figure 1 and are described below:

- **Basic system (BS)**. For this experiment, we have combined the 3-turn conversations of the corpus in a text string separated by spaces. For example, for turn 1: ”Hahah i loved it” , turn 2: ”Yay! Glad you loved it X3” and turn 3: ”You always make us happy”, the final sentence is ”Hahah i loved it Yay! Glad you loved it X3 You always make us happy”. Then, each sentence is represented as a vector of unigrams and bigrams choosing the TF weighting scheme and it is used as feature for the classification using the SVM algorithm.

- **Basic system with turn 1 and 2 (BS-2)**. This experiment is similar to the previous one. However, we have only taken into account the first and last conversation turns because analyzing the training data, we realized that the second conversation turn is not useful for the classification as it does not provide representative information. For example, for turn 1: ”Hahah i loved it” , turn 2: ”Yay! Glad you loved it X3” and turn 3: ”You always make us happy”, the final sentence is ”Hahah i loved it You always make us happy”. We notice that the emotion is the same (happy) as if we consider the three turns.

- **System with features (SF)**. In this system, also we have only taken into account the first and last conversation turns. With these turns of conversations, we have tested several combinations with the lexical resources during the development phase and we chose the best combination for the evaluation phase. The best combination is the set of the vector of NRC (four values), the vector of Emolex (eight values) and the vector of VaderSentiment (four values) explained in Subsection 4.2. Therefore, the union of the best lexical features and the TF of the two conversation turns are used as features to perform the classification with the selected SVM algorithm.

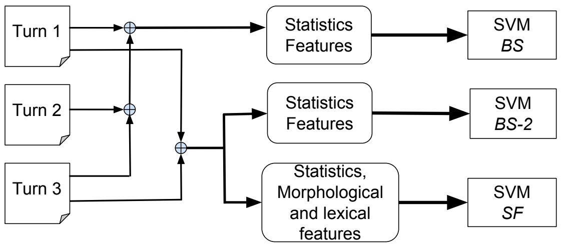


Figure 1: Systems architecture.

The official competition metric to evaluate the

[2]https://www.nltk.org/_modules/nltk/sentiment/vader.html

Experiment	Sad			Angry			Happy			Micro - Avg		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
BS	0.49	0.73	0.58	0.54	0.79	0.64	0.43	0.73	0.54	0.48	0.75	0.59
BS-2	0.64	0.74	0.68	0.60	0.85	0.70	0.57	0.76	0.65	0.60	0.79	0.68
SF	0.63	0.81	0.71	0.62	0.87	0.72	0.57	0.77	0.66	0.61	0.82	0.7

Table 2: Results on the dev set.

Experiment	Sad			Angry			Happy			Micro - Avg		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
BS	0.59	0.74	0.66	0.53	0.77	0.66	0.5	0.64	0.56	0.56	0.72	**0.63**
BS-2	0.64	0.77	0.7	0.59	0.8	0.68	0.62	0.71	0.66	0.61	0.76	**0.68**
SF	0.61	0.78	0.69	0.58	0.81	0.68	0.61	0.70	0.65	0.6	0.76	**0.67**

Table 3: Results on the test set.

User name (ranking)	F1
leo1020 (1)	0.79
gautam_naik (60)	0.72
fmplaza (92)	**0.68**
emocontext_organizers (140)	0.59
waylensu (161)	0.0143

Table 4: System test results per user in EmoContext task.

systems in EmoContext task is the microaveraged F1 score (F1μ) for the three emotion classes (Happy, Sad and Angry). This metric is calculated between the real classes and the predicted classes. The results of our participation in the task can be seen in Tables 2 and 3.

In relation to our results, during the pre-evaluation phase and evaluation phase, we noticed that 1 and 3 conversation turns performed better the classification due to the reason that the 2 conversation turn is usually a contradiction or a question of the 1-turn. In Tables 2 and 3 we can observed that the BS-2 experiments outperformed the BS experiments. According to the classification per emotion, we may note different issues. On the one hand, the use of lexical features (SF experiment) improve about 2% of F1 with respect to the BS-2 experiment in the dev set. Nevertheless, this is not the case in the test set. On the other hand, the Happy emotion class perform worse than other emotion classes in both datasets, as it happens in other works (Chatterjee et al., 2019a; Gupta et al., 2017). Besides, if we observed the SF experiment in test set, we can see that the emotional features do not help to improve the classification because there are some words like “love” or “cool” whose assigned emotion is Happy class but in the 3-turn conversation of test set have been marked as Others class by the judges. Finally, in Table 4 we can observe our official position in the competition. We are ranked 92 out of 165 participating teams and our system outperforms the baseline system provided by the organizers of the task.

6 Conclusions and Future Work

In this paper, we present different systems to predict the emotion of user in a textual dialogue along with two turns of context. To carry out the task, we have developed three different systems. The first two are base systems, combining different turns of conversation and in the last system we decided to incorporate lexical features from sentiment and emotional resources.

In the future, we plan to continue working in emotion classification tasks because we have observed that the participation in this tasks is very high and this shows the interest by the scientific community in solving this type of tasks. Efforts will also be made to include more contextual information and to explore other multiple classifier methods.

Acknowledgments

This work has been partially supported by Fondo Europeo de Desarrollo Regional (FEDER) and REDES project (TIN2015-65136-C2-1-R) from the Spanish Government.

References

Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. 2019a. Understanding emotions in text using deep learning and big data. *Computers in Human Behavior*, 93:309–317.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019b. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

CJ Hutto Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14). Available at (20/04/16) http://comp. social. gatech. edu/papers/icwsm14. vader. hutto. pdf.*

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations. *arXiv preprint arXiv:1707.06996*.

Nida Manzoor Hakak, Mohsin Mohd, Mahira Kirmani, and Mudasir Mohd. 2017. Emotion analysis: A survey. In *Computer, Communications and Electronics (Comptelix), 2017 International Conference on*, pages 397–402. IEEE.

Saif M Mohammad. 2017. Word affect intensities. *arXiv preprint arXiv:1704.08798*.

Saif M Mohammad and Peter D Turney. 2010. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*, pages 26–34. Association for Computational Linguistics.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Jean Thilmany. 2007. The emotional robot: Cognitive computing and the quest for artificial intelligence. *EMBO reports*, 8(11):992–994.

Ali Yadollahi, Ameneh Gholipour Shahraki, and Osmar R Zaiane. 2017. Current state of text sentiment analysis from opinion to emotion mining. *ACM Computing Surveys (CSUR)*, 50(2):25.

SNU_IDS at SemEval-2019 Task 3: Addressing Training-Test Class Distribution Mismatch in Conversational Classification

Sanghwan Bae, Jihun Choi and **Sang-goo Lee**
Department of Computer Science and Engineering
Seoul National University, Seoul, Korea
{sanghwan,jhchoi,sglee}@europa.snu.ac.kr

Abstract

We present several techniques to tackle the mismatch in class distributions between training and test data in the Contextual Emotion Detection task of SemEval 2019, by extending the existing methods for class imbalance problem. Reducing the distance between the distribution of prediction and ground truth, they consistently show positive effects on the performance. Also we propose a novel neural architecture which utilizes representation of overall context as well as of each utterance. The combination of the methods and the models achieved micro F1 score of about 0.766 on the final evaluation.

1 Introduction

A new task whose goal is to predict the emotion of the last speaker given a sequence of text messages has been designed, and coined Contextual Emotion Detection (EmoContext; Chatterjee et al., 2019).

Though predicting the emotion of a single utterance or sentence, *i.e.* emotion detection, is a well-discussed subject in natural language understanding literature, EmoContext has several novel challenges. Firstly, the class distribution of training data is significantly different from that of the test data. Consequently, a model trained on the training data might not perform well on the test data. There have been efforts made to address the problem of learning from training data sets that have imbalanced class distribution, i.e. the class imbalance problem (Chawla et al., 2004; Buda et al., 2018). However, they are not applicable to our case, since the imbalance appears only in the test data while the training data can be viewed to be balanced.

We extend the existing methods of addressing the class imbalance problem to be applicable for more general cases where the distributions of the collected training data differ from those of the real population or the data at test time, under the assumption that the validation set is organized carefully to reflect the practical distribution. From experiments and analyses, we show that the proposed methods reduce the difference between two distributions and as a result improve the performance of the model.

The additional challenge we have to consider arises from the fact that utterances having identical surface form may have different meanings due to sarcasm, irony, or etc.. Thus a model should track the emotional transitions within a dialogue. To grasp the context of utterances, we propose a semi-hierarchical encoder structure. Lastly, the texts contain lots of non-standard words, *e*.g. emoticons and emojis. This makes it difficult to exploit typical pre-trained embeddings such as GloVe (Pennington et al., 2014). Therefore, we adopt pre-trained embeddings which are specialized for handling non-standard words. We show that the proposed model largely outperforms the baseline of task organizers by experiments.

2 Related Work

2.1 Contextual Emotion Detection

EmoContext is a emotion classification data set composed of 3-turn textual conversations. The goal of the data set is to classify the emotion in the last utterance of each example given the context. The label set consists of 4 classes: 'happy', 'sad', 'angry' and 'others'. In the training data set, there are about 50% of 'others' class examples and 50% of emotional (happy, sad, angry) examples, which can be viewed as well-balanced. On the other hand, only 15% of examples in the test and the validation data set are labeled as emotional, reflecting the real-life frequency. For more details, refer to Chatterjee et al. (2019).

2.2 Class Imbalance Problem

When some classes have the significantly higher number of examples than other classes in a training set, the data is said to have an imbalanced class distribution (Buda et al., 2018). This makes it difficult to learn from the data set using machine learning approaches (Batista et al., 2004; Mazurowski et al., 2008), since the learned models can be biased to majority classes easily, which results in poor performance (Wang et al., 2016). We give a brief explanation of methods to solve this problem.

Sampling: This type of methods deals with the problem by manipulating the data itself to make the resulting data distribution balanced. The simplest versions are *random oversampling* and *random undersampling*. The former randomly duplicates examples from the minority classes and the latter randomly removes instances from the majority classes (Mollineda et al., 2007).

Thresholding: This method moves the decision threshold after training phase, changing the output class probabilities. Typically, this can be done by simply dividing the output probability for each class by its estimated prior probability (Richard and Lippmann, 1991; Buda et al., 2018).

Cost-Sensitive Learning: This assigns different misclassification cost for each class and applies the cost in various ways, *e.g.* output of the network, learning rate or loss function. For multi-class classification tasks, the simplest form of the cost-sensitive learning is to introduce weights to the cross entropy loss (Han, 2017; Lin et al., 2018):

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{K} w^c y_i^c \ln p(c|x_i), \qquad (1)$$

where N and K denote the total number of examples and classes respectively, $p(c|x_i)$ the predicted probability of i-th example x_i belonging to class c, y_i^c the ground truth label which is 0 or 1, and w^c a class dependent weighting factor. Recent work suggests to use the inverse ratio of each class, *i.e.* $w^c = \frac{N}{N_c}$, where N_c is the number of examples belonging to class c (Wu et al., 2018; Aurelio et al., 2019).

Ensemble: The term ensemble usually refers to a collection of classifiers that are minor variants of the same classifier to boost the performance. This is also successfully applied to the class imbalance problem (Sun et al., 2007; Seiffert et al., 2010), by replacing the resampling procedure in the bagging algorithm with oversampling or undersampling (Galar et al., 2012).

3 Methods for Mismatch Problem

3.1 Sampling

In our case, it is not possible to make the distinction between majority and minority classes; even if the 'others' class is the most prevalent in the training data, the ratio is less than that of the test data. To address this issue, the random minority oversampling technique should be modified, since it assumes that the class imbalance appears only in the training data set. Accordingly, we apply random oversampling or random undersampling to make the distribution of the training data similar to that of the validation data.

3.2 Thresholding

The basic thresholding method mentioned in §2.2 is not sufficient for our case, since we should additionally *bias* the model output distribution to match the test time distribution, not only correcting the imbalance in the training data. When p_r is a probability of training time and p_s is that of validation time, we multiply the predicted probability by the estimated class ratio from validation set as below:

$$\begin{aligned} y_c(x) = p(c|x) &\approx \frac{\boldsymbol{p_s(c)}}{p_r(c)} \cdot p_r(c|x) \\ &= \frac{\boldsymbol{p_s(c)}}{p_r(c)} \cdot \frac{p_r(c)\cdot p_r(x|c)}{p_r(x)} \qquad (2) \\ &= \frac{\boldsymbol{p_s(c)}\cdot p_r(x|c)}{p_r(x)}, \end{aligned}$$

where $p(c) = \frac{N_c}{N}$.

3.3 Cost-Sensitive Learning

The weighted cross entropy loss, described in Eq. (1), can be used for our case, as long as the weights are carefully chosen. Although the reciprocal of class ratio is helpful for learning balanced prediction (Wu et al., 2018), the target distribution between classes is not uniform for our task. Also, the misclassification cost of 'others' class should be larger than those of emotional classes, because the model tends to predict it less than the actual. Therefore, it is reasonable to modify w^c by multiplying the estimated ratio of each class for test time, *i.e.* $w^c = \frac{N^r}{N_c^r} \cdot \frac{N_c^s}{N^s}$, where N^r and N^s denote

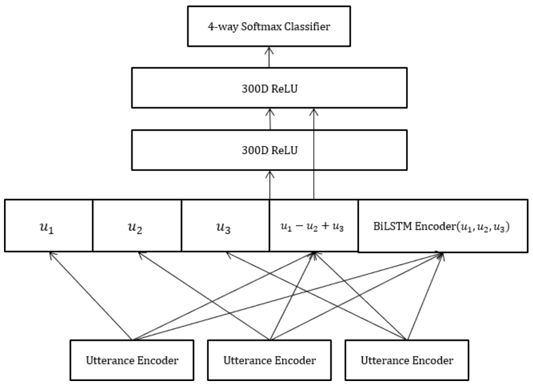


Figure 1: Overall Architecture

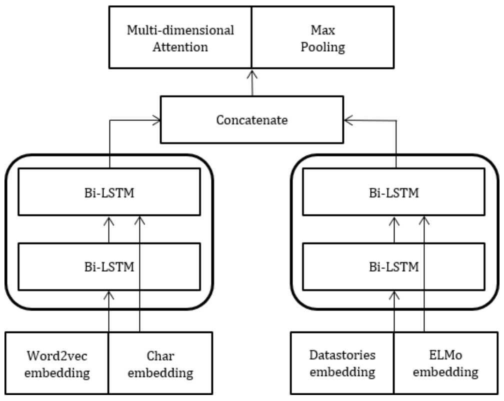


Figure 2: Utterance Encoder

the number of instances in training and validation data set respectively. This corresponds to the term introduced in Eq. (2), as $\frac{N^r}{N^r_c} \cdot \frac{N^s_c}{N^s} = \frac{p_s(c)}{p_r(c)}$. The difference between them is that thresholding utilizes the term in inference phase after training is finished, while weighted cross entropy loss includes it from the training time.

3.4 Ensemble

We combined bagging-based ensemble with sampling techniques represented in §3.1. In addition, we compared ensembles of randomly initialized classifiers using other methods (*i.e.* thresholding and cost-sensitive learning).

4 Model and Training Details[1]

4.1 Overall Architecture

We propose a semi-hierarchical structure to capture the context as well as the meaning of each single utterance. Fig. 1 depicts the overall architecture. Each single utterance representation u_m is encoded by Utterance Encoder described in Fig. 2. In addition, we introduce another bi-directional LSTM (BiLSTM) encoder for higher level representation which receives the outputs of utterance encoder as its inputs. The representation of all context is generated by the concatenation of u_1, u_2, u_3, $u_1 - u_2 + u_3$ and the output of this encoder. Then it is fed to the two 300-dimensional (300D) hidden layers with $ReLU$ activation with shortcut connections and a $softmax$ output layer.

4.2 Utterance Encoder

The proposed utterance encoder has two types of shortcut-stacked bi-directional long short-term memory (BiLSTM) encoder (Nie and Bansal, 2017) to fully exploit four types of lexicon level representations. The first encoder utilizes pre-trained word2vec (Mikolov et al., 2013) embeddings concatenated with trainable embeddings from a character level convolutional neural network (CNN). We also added emoji2vec (Eisner et al., 2016) embeddings to word2vec to map emoji symbols to the same space. The other encoder receives concatenated representations of pre-trained datastories (Baziotis et al., 2017) embeddings and contextualized represenations from ELMo (Peters et al., 2018).

The results from the two stacked BiLSTM encoder are concatenated. We use the multi-dimensional source2token self-attention of Shen et al. (2018) and max pooling to integrate contextualized word level representations to a single utterance representation, as in Im and Cho (2017).

4.3 Training Details

We use 300D Google News word2vec[2] embeddings and 300D pre-trained emoji2vec.[3] Datastories vectors[4] which were pre-trained on a big collection of Twitter messages using GloVe are also 300D. The dimension of character embeddings is fixed to 15, and it is fed to a CNN where the filter sizes are 1, 3, and 5 and the number of feature map for each is 100, thus a 300D vector is generated for each word as a result. To guarantee the same size for ELMo embeddings, a 300D position-wise feed-forward network is applied above them. The hidden states of all the BiLSTMs for each direc-

[1]The implementation of our model is available at https://github.com/baaesh/semeval19_task3

[2]https://code.google.com/archive/p/word2vec/

[3]https://github.com/uclmr/emoji2vec

[4]https://github.com/cbaziotis/datastories-semeval2017-task4

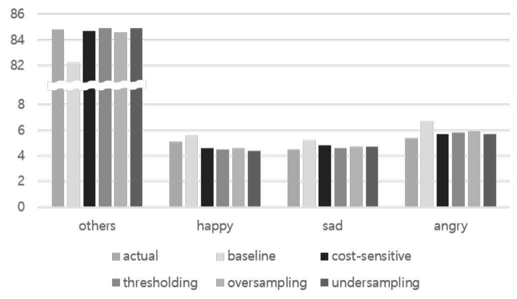


Figure 3: Class distributions on test data. Actual is from the ground truth labels and the others are from the predicted labels by each model. These are the averages of 10 results with random initialization.

tion are 150D and the number of layers is 2.

Our model is trained using the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.001 and a batch size 64. We clip the norm of gradients to make it smaller than 3. Dropout (Srivastava et al., 2014) technique is applied to word embeddings with $p = 0.1$. We chose the best model based on a micro F1 score on the validation set.

5 Experiments

5.1 Effects of mismatch in class distributions

Fig. 3 shows that the class distribution of predictions from the baseline model without any methods applied is substantially different from that of the actual test data. On the other hand, when the proposed methods are applied, the gap becomes much smaller. From this result, we conjecture that the difference in output distributions could be a reason for poor performance of the baseline compared to the proposed methods, as presented in Table 1.

5.2 Single model methods

Table 1 shows the accuracy and micro F1 scores of variants of our methods and baselines on the test set. We report the mean and standard deviation of 10 experimental runs (with the same hyperparameters) for each methods. And all the models were chosen based on their performance on the validation set.

The reported results show that proposed methods except undersampling are effective for enhancing both accuracy and F1 score. This means that alleviating the difference of class distribution is the key factor for the higher performance. In the case of undersampling, since the total size of training data decreases, the model seems to fail to capture the general semantics. The result is consistent with Buda et al. (2018), where the undersampling solely does not bring a performance gain for deep learning models. On the other hand, in our experiments, thresholding and cost-sensitive learning were the most effective approaches when a single model is used.

Approach	**Acc**	(±)	**F1**	(±)
Baseline (organizers)	-	-	.587	-
Baseline (ours)	.914	.005	.726	.008
Oversampling	.922	.004	.733	.012
Undersampling	.919	.006	.719	.013
Thresholding	**.924**	.002	.738	.010
Cost-Sensitive	.924	.004	**.739**	.010

Table 1: Comparison of single model approaches on the test set.

Approach	**Acc**	**F1**
Baseline (ours)	.921	.743
Oversampling	.930	**.758**
Undersampling	.930	.753
Thresholding	.930	.752
Cost-Sensitive	**.931**	.757
Mixed (submitted)	**.933**	**.766**

Table 2: Comparison of ensemble approaches on the test set. 10 models were used for each ensemble result.

5.3 Ensemble methods

Table 2 reports the comparison of ensemble models. The results show that our methods consistently outperform the baseline. We can see that ensemble with bagging has a great effect on refining class distribution, and in this time, undersampling also showed a good performance. Overall, for ensemble methods, oversampling and cost-sensitive learning performed best. The version we submitted to the leaderboard was the ensemble of different methods selected by their performance on validation set, and achieved the official result of 0.766.

6 Conclusion

In this paper, we proposed several methods for alleviating the problems caused by difference in class distributions between training data and test data. We demonstrated that these methods have positive effects on the result performance. We also presented a novel semi-hierarchical neural architecture that effectively exploits the context and the utterance representation. For future work, we plan to conduct more systematic experiments on other data sets to generalize our results.

References

Yuri Sousa Aurelio, Gustavo Matheus de Almeida, Cristiano Leite de Castro, and Antonio Padua Braga. 2019. Learning from imbalanced data sets with weighted cross-entropy function. *Neural Processing Letters*, pages 1–13.

Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. 2004. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, 6(1):20–29.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754.

Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. 2018. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

NV Chawla, N Japkowicz, and A Kotcz. 2004. Editorial: special issue on learning from imbalanced data sets. sigkdd explor newsl 6: 1–6.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. *arXiv preprint arXiv:1609.08359*.

Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. 2012. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484.

Xiao Han. 2017. Automatic liver lesion segmentation using a deep convolutional neural network method. *arXiv preprint arXiv:1704.07239*.

Jinbae Im and Sungzoon Cho. 2017. Distance-based self-attention network for natural language inference. *arXiv preprint arXiv:1712.02047*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2018. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*.

Maciej A Mazurowski, Piotr A Habas, Jacek M Zurada, Joseph Y Lo, Jay A Baker, and Georgia D Tourassi. 2008. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural networks*, 21(2-3):427–436.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

RA Mollineda, R Alejo, and JM Sotoca. 2007. The class imbalance problem in pattern classification and learning. In *II Congreso Espanol de Informática (CEDI 2007). ISBN*, pages 978–84.

Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. *arXiv preprint arXiv:1708.02312*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Michael D Richard and Richard P Lippmann. 1991. Neural network classifiers estimate bayesian a posteriori probabilities. *Neural computation*, 3(4):461–483.

Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. 2010. Rusboost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 40(1):185–197.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Yanmin Sun, Mohamed S Kamel, Andrew KC Wong, and Yang Wang. 2007. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378.

Shoujin Wang, Wei Liu, Jia Wu, Longbing Cao, Qinxue Meng, and Paul J Kennedy. 2016. Training deep neural networks on imbalanced data sets. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 4368–4374. IEEE.

Zhenyu Wu, Yang Guo, Wenfang Lin, Shuyang Yu, and Yang Ji. 2018. A weighted deep representation learning model for imbalanced fault diagnosis in cyber-physical systems. *Sensors*, 18(4):1096.

SSN_NLP at SemEval-2019 Task 3: Contextual Emotion Identification from Textual Conversation using Seq2Seq Deep Neural Network

B. Senthil Kumar, D. Thenmozhi, Chandrabose Aravindan, S. Srinethe
Department of CSE, SSN College of Engineering, India
{senthil,theni_d,aravindanc}@ssn.edu.in
srinethe16108@cse.ssn.edu.in

Abstract

Emotion identification is a process of identifying the emotions automatically from text, speech or images. Emotion identification from textual conversations is a challenging problem due to absence of gestures, vocal intonation and facial expressions. It enables conversational agents, chat bots and messengers to detect and report the emotions to the user instantly for a healthy conversation by avoiding emotional cues and miscommunications. We have adopted a Seq2Seq deep neural network to identify the emotions present in the text sequences. Several layers namely embedding layer, encoding-decoding layer, softmax layer and a loss layer are used to map the sequences from textual conversations to the emotions namely Angry, Happy, Sad and Others. We have evaluated our approach on the EmoContext@SemEval2019 dataset and we have obtained the micro-averaged F1 scores as 0.595 and 0.6568 for the pre-evaluation dataset and final evaluation test set respectively. Our approach improved the base line score by 7% for final evaluation test set.

1 Introduction

Emotion identification is a process of identifying the emotions automatically from different modalities. Several research work have been presented on detecting emotions from text (Rao, 2016; Abdul-Mageed and Ungar, 2017; Samy et al., 2018; Al-Balooshi et al., 2018; Gaind et al., 2019), speech (Arias et al., 2014; Amer et al., 2014; Lim et al., 2016), images (Shan et al., 2009; Ko, 2018; Ayvaz et al., 2017; Faria et al., 2017; Mohammadpour et al., 2017) and video (Matsuda et al., 2018; Hossain and Muhammad, 2019; Kahou et al., 2016). Emotion understanding from video may be easier by analyzing the body language, speech variations and facial expressions. However, identification of emotions from textual conversations is a challenging problem due to absence of above factors. Emotions in text are not only identified by its cue words such as *happy, good, bore, hurt, hate* and *fun*, but also the presence of interjections (e.g. "whoops"), emoticons (e.g. ":)"), idiomatic expressions (e.g. "am in cloud nine"), metaphors (e.g. "sending clouds") and other descriptors mark the existence of emotions in the conversational text. Recently, the growth of text messaging applications for communications require emotion detection from conversation transcripts. This helps conversational agents, chat bots and messengers to avoid emotional cues and miscommunications by detecting the emotions during conversation. EmoContext@SemEval2019 shared task (Chatterjee et al., 2019) goal is to encourage more research in the field of contextual emotion detection in textual conversations. The shared task focuses on identifying emotions namely Angry, Happy, Sad and Others from conversation with three turns. Since, emotion detection is a classification problem, research works have been carried out by using machine learning with lexical features (Sharma et al., 2017) and deep learning with deep neural network (Phan et al., 2016) and convolutional neural network (Zahiri and Choi, 2018) to detect the emotions from text. However, we have adopted Seq2Seq deep neural network for detecting the emotions from textual conversations which include sequence of phrases. This paper elaborates our Seq2Seq approach for identifying emotions from text sequences.

2 Related Work

This section reviews the research work reported for emotion detection from text / tweets (Perikos and Hatzilygeroudis, 2013; Rao, 2016; Abdul-Mageed and Ungar, 2017; Samy et al., 2018; Al-Balooshi et al., 2018; Gaind et al., 2019) and text conversations (Phan et al., 2016; Sharma et al., 2017; Zahiri and Choi, 2018).

Sharma et al. (2017) proposed a methodology to create a lexicon - a vocabulary consisting of positive and negative expressions. This lexicon is used to assign an emotional value which is derived from a fuzzy set function. Gaind et al. (2019) classified twitter text into emotion by using textual and syntactic features with SMO and decision tree classifiers. The tweets are annotated manually by Liew and Turtle (2016) with 28 fine-grained emotion categories and experimented with different machine learning algorithms. Results show that SVM and BayesNet classifiers produce consistently good performance for fine-grained emotion classification. Phan et al. (2016) developed an emotion lexicon from WordNet. The conversation utterances are mapped to the lexicons and 22 features are extracted using rule-based algorithm. They used fully connected deep neural network to train and classify the emotions. TF-IDF with handcrafted NLP features were used by Al-Balooshi et al. (2018) in logistic regression, XGBClassifier and CNN+LSTM for emotion classification. The authors found that the logistic regression performed better than the deep neural network model. All the models discussed above considered the fine-grained emotion categories and used the twitter data to create a manually annotated corpus. These models used the rule-based or machine learning based algorithms to classify the emotion category.

A new C-GRU (Context-aware Gated Recurrent Units) a variant of LSTM was proposed by Samy et al. (2018) which extracts the contextual information (topics) from tweets and uses them as an extra layer to determine sentiments conveyed by the tweet. The topic vectors resembling an image are fed to CNN to learn the contextual information. Abdul-Mageed and Ungar (2017) built a very large dataset with 24 fine-grained types of emotions and classified the emotions using gated RNN. Instead of using basic CNN, a new recurrent sequential CNN is used by Zahiri and Choi (2018). They proposed several sequence-based convolution neural network (SCNN) models with attention to facilitate sequential dependencies among utterances. All the models discussed above show that the emotion prediction can be handled using variants of deep neural network such as C-GRU, G-RNN and Sequential-CNN. The commonality between the above models are the variations of RNN or LSTM. This motivated us to use the Sequence-to-Sequence (Seq2Seq) model which consists of stacked LSTMs to predic the emotion labels conditioned on the given utterance sequences.

3 Data and Preprocessing

We have used the dataset provided by EmoContext@SemEval2019 shared task in our approach. The dataset consists of training set, development set and test set with 30160, 2755 and 5509 instances respectively. The dataset contains sequence id, text sequences with three turns which include user utterance along with the context, followed by emotion class label. The task is to label the user utterance as one of emotion class: happy, sad, angry or others. The textual sequences contain many short words. In preprocessing, these words are replaced with original or full word. We resort to build a look-up table which replace ''m', with 'am', ''re' with 'are', ''ere' with 'were', 'n't' with 'not', ''ll' with 'will', ''d' with 'would', 'what's' with 'what is' and 'it's' with 'it is'. The sequences are converted to lower case. Also, the three turns/sentences are delimited with "eos" in the input sequences.

4 Methodology

Seq2Seq model is the most popular model in learning the target sequence conditioned on the source sequence. The Seq2Seq model is adopted to map the sequences of n words with a target label (n:1 mapping). This model has an embedding layer, an encoder, a decoder and a projection layer as shown in Figure 1.

Once the dialogue sentences are preprocessed, the first three turns of each instance are considered as the input sequences w_1, w_2,..,w_n, and the corresponding label e is considered as the target sequence. For example, the given instance "13 Bad Bad bad! That's the bad kind of bad. I have no gf sad" is converted into input sequence "bad eos bad bad that is the bad kind of bad eos i have no gf" and target label "sad". The input sequences and the target label are converted into its corresponding word embeddings by the embedding layer. The vector representation for each word is derived at embedding layer by choosing a fixed vocabulary of size V for input sequences and target labels.

Now, the encoder which uses Bi-LSTM, encode these embeddings into a fixed vector representa-

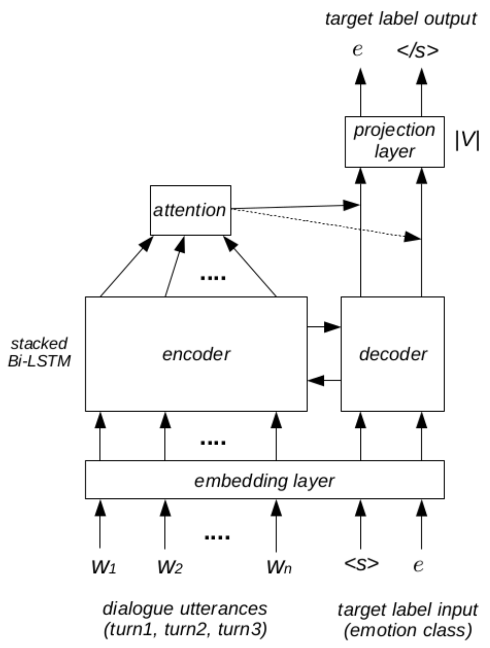


Figure 1: System Architecture

tion s which also represents the summary of input sequences. Once the source sequences are encoded, the last hidden state of the encoder is used to initialize the decoder. The projection layer is fed with the tensors of the target output label. Given the hidden state h_t , the decoder predicts the label e_t. However, h_t and e_t are conditioned on the previous output e_{t-1} and on the summary s of the input sequence. The projection layer is a dense matrix to turn the top hidden states of decoder to logit vectors of dimension V. Given the logit values, the training loss is easily minimized by using standard SGD optimizer with a learning rate. The model is also trained with the attention mechanism, which computes the attention weight by comparing the current decoder hidden state with all encoder states. The detailed description of working principle about Seq2Seq model is described in (Sutskever et al., 2014).

We have adopted Neural Machine Translation[1] code to implement our Seq2Seq deep neural network. Several variations have been implemented by varying the number of layers, units and attention mechanisms. It is evident from the earlier experiments (Sutskever et al., 2014; Thenmozhi et al., 2018) that bi-directional LSTM performs better for short text sequences. Hence, we have used it for encoding and decoding processes. The models were trained for 30000 steps with drop out of 0.2. We have utilized two attention wrappers namely Normed_Bahdanau (NB) (Sutskever et al., 2014; Bahdanau et al., 2014) and Scaled_Luong (SL) (Luong et al., 2015, 2017).

Since, the model was developed using deep learning technique, it does not require much of linguistic features such as stemming, case normalization and PoS in identifying the emotion cue words. These linguistic phenomena could be captured by the encoder RNNs in sequence-to-sequence (Seq2Seq) model. The other statistical features such as the word frequency are also not considered as input to the model, because the presence of particular cue alone does not guarantee to detect emotions in the text.

[1]https://github.com/tensorflow/nmt

Models	**F1 μ Score**
8L_SL_No_split	0.523
8L_NB_No_split	0.527
8L_NB_TV_split	0.5296
8L_NB_EOS_TV_split	0.5499
16L_NB_No_split	0.510
16L_NB_TV_split	0.526
16L_NB_EOS_TV_split	0.547
32L_NB_EOS_No_split	0.531
32L_NB_EOS_TV_split	0.5398
2L_NB_EOS_No_split	0.544
2L_NB_EOS_TV_split	**0.595**

Table 1: Development Set Micro-avereged F1 Score.

5 Results

Our approach is evaluated on EmoContext@SemEval2019 data set. During development, we have implemented our variations with and without end of sentence (EOS) delimiter. We have built the models using entire training set (No_split) and train-validation splits (TV_split). 27160 and 3000 instances from training data were considered as training and validation set in TV_split. The performance was measured in terms of micro-averaged F1 score (F1μ) for the three emotion classes namely Angry, Happy and Sad.

We have submitted eleven runs for EmoContext@SemEval2019 shared task on pre-evaluation dataset. The results obtained for pre-evaluation dataset are given in Table 1.

We observe from Table 1 that Normed_Bahdanau attention mechanism performs better than Scaled_Luong. Model building

Models	**F1 μ Score**
16U_TV_split_1	0.649422
32U_TV_split_1	0.416399
64U_TV_split_1	**0.656752**
128U_TV_split_1	0.626124
256U_TV_split_1	0.581599
16U_TV_split_2	0.59668
32U_TV_split_2	0.617944
64U_TV_split_2	0.618201
128U_TV_split_2	0.622652
256U_TV_split_3	0.642144
16U_TV_split_3	0.611716
32U_TV_split_3	0.567093
64U_TV_split_3	0.624924
128U_TV_split_3	0.655106
256U_TV_split_3	0.612288

Table 2: Final Evaluation Test Data Micro-avereged F1 Score .

Models	**F1 Score**		
	Happy	**Sad**	**Angry**
16U_TV_split_1	0.619	0.645	0.686
32U_TV_split_1	0.299	0.550	0.384
64U_TV_split_1	**0.633**	0.638	**0.695**
128U_TV_split_1	0.606	0.583	0.689
256U_TV_split_1	0.553	0.566	0.626
16U_TV_split_2	0.525	0.584	0.684
32U_TV_split_2	0.537	0.637	0.677
64U_TV_split_2	0.585	0.615	0.657
128U_TV_split_2	0.596	0.595	0.676
256U_TV_split_3	0.609	0.637	0.681
16U_TV_split_3	0.513	0.641	0.679
32U_TV_split_3	0.507	0.588	0.607
64U_TV_split_3	0.552	0.637	0.685
128U_TV_split_3	0.612	**0.664**	0.692
256U_TV_split_3	0.559	0.618	0.657

Table 3: Class-wise F1 Score for Final Evaluation Test Data.

with TV_split performs better than the model without split. The incorporation of delimiter text EOS also improved the performance of our approach. Further, the performance degrades with the increase in number of layers. Thus, 2 layered LSTM with TVsplit, EOS delimiter and Normed_Bahdanau attention mechanism perform better on the pre-evaluation dataset of EmoContext@SemEval2019 and this architecture is considered for evaluating the final-evaluation test set. The final evaluation submissions are based upon the variations in TV_split ratio and the number of units as 16, 32, 64, 128 and 256. For TV_split_1, the development set (2755 instances) given by EmoContext@SemEval2019 was considered as a validation set. The other two TV_splits are by keeping the validation set as 1/5 (TV_split_2) and 1/3 (TV_split_3) of training set. The results of our submissions on final evaluation test data are given in Table 2. It is observed from Table 2 that 64U_TV_split_1 model outperforms all the other models with 0.656752 F1μ score. This score is higher than the base line score with 7% improvement. Table 3 shows the class-wise performance of our models on final evaluation set. Our models perform better for Angry class than the other two classes namely Happy and Sad.

6 Conclusion

We have adopted a Seq2Seq deep neural network to identify the emotions present in the text sequences. Our approach is evaluated on the EmoContext@SemEval2019 dataset. The input sequences are pre-processed by replacing the short hand notations and by introducing a delimiter string. The sequence is vectorized using word embeddings and given to bi-directional LSTM for encoding and decoding. We have implemented several variations by changing the parameters namely, number of layers, units, attention wrappers, with and without delimiter string and train-validation split. The performance is measured using micro-averaged F1 score on three emotion class labels namely Angry, Happy and Sad. Our experiments on development set show that 2 layered LSTM with Normed_Bahdanau attention mechanism with delimiter string and train-validation split performs better than all the other variations. Three variations of train-validation split ratio were experimented on final evaluation test data by varying the number of units with the best parameter values that are learnt during the development phase. 64U_TV_split_1 model performs better than all the other runs we have submitted to the task. This model shows 7% improvement than the base line on final evaluation test set. Our Seq2Seq model can be improved further by incorporating the soft attention mechanism which uses joint distribution between attention and output layer (Shankar et al., 2018).

References

Muhammad Abdul-Mageed and Lyle Ungar. 2017. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 718–728.

Hessa AlBalooshi, Shahram Rahmanian, and Rahul Venkatesh Kumar. 2018. Emotionx-smartdubai_nlp: Detecting user emotions in social media text. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, pages 45–49.

Mohamed R Amer, Behjat Siddiquie, Colleen Richey, and Ajay Divakaran. 2014. Emotion detection in speech using deep networks. In *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 3724–3728. IEEE.

Juan Pablo Arias, Carlos Busso, and Nestor Becerra Yoma. 2014. Shape-based modeling of the fundamental frequency contour for emotion detection in speech. *Computer Speech & Language*, 28(1):278–294.

Uğur Ayvaz, Hüseyin Gürüler, and Mehmet Osman Devrim. 2017. Use of facial emotion recognition in e-learning systems. *Information Technologies and Learning Tools*, 60(4):95–104.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *In Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Ana Raquel Faria, Ana Almeida, Constantino Martins, Ramiro Gonçalves, José Martins, and Frederico Branco. 2017. A global perspective on an emotional learning model proposal. *Telematics and Informatics*, 34(6):824–837.

Bharat Gaind, Varun Syal, and Sneha Padgalwar. 2019. Emotion detection and analysis on social media. *arXiv preprint arXiv:1901.08458*.

M Shamim Hossain and Ghulam Muhammad. 2019. Emotion recognition using deep learning approach from audio–visual emotional big data. *Information Fusion*, 49:69–78.

Samira Ebrahimi Kahou, Xavier Bouthillier, Pascal Lamblin, Caglar Gulcehre, Vincent Michalski, Kishore Konda, Sébastien Jean, Pierre Froumenty, Yann Dauphin, Nicolas Boulanger-Lewandowski, et al. 2016. Emonets: Multimodal deep learning approaches for emotion recognition in video. *Journal on Multimodal User Interfaces*, 10(2):99–111.

Byoung Ko. 2018. A brief review of facial emotion recognition based on visual information. *sensors*, 18(2):401.

Jasy Suet Yan Liew and Howard R Turtle. 2016. Exploring fine-grained emotion detection in tweets. In *Proceedings of the NAACL Student Research Workshop*, pages 73–80.

Wootaek Lim, Daeyoung Jang, and Taejin Lee. 2016. Speech emotion recognition using convolutional and recurrent neural networks. In *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 1–4. IEEE.

Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. 2017. Neural machine translation (seq2seq) tutorial. *https://github.com/tensorflow/nmt*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Yuki Matsuda, Dmitrii Fedotov, Yuta Takahashi, Yutaka Arakawa, Keiichi Yasumoto, and Wolfgang Minker. 2018. Emotour: Multimodal emotion recognition using physiological and audio-visual features. In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, pages 946–951. ACM.

Mostafa Mohammadpour, Hossein Khaliliardali, Seyyed Mohammad R Hashemi, and Mohammad M AlyanNezhadi. 2017. Facial emotion recognition using deep convolutional networks. In *2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, pages 0017–0021. IEEE.

Isidoros Perikos and Ioannis Hatzilygeroudis. 2013. Recognizing emotion presence in natural language sentences. In *International conference on engineering applications of neural networks*, pages 30–39. Springer.

Duc Anh Phan, Hiroyuki Shindo, and Yuji Matsumoto. 2016. Multiple emotions detection in conversation transcripts. In *Proceedings of the 30th Pacific Asia Conference on Language, Information and Computation: Oral Papers*, pages 85–94.

Yanghui Rao. 2016. Contextual sentiment topic model for adaptive social emotion classification. *IEEE Intelligent Systems*, 31(1):41–47.

Ahmed E Samy, Samhaa R El-Beltagy, and Ehab Hassanien. 2018. A context integrated model for multi-label emotion detection. *Procedia computer science*, 142:61–71.

Caifeng Shan, Shaogang Gong, and Peter W McOwan. 2009. Facial expression recognition based on local binary patterns: A comprehensive study. *Image and vision Computing*, 27(6):803–816.

Shiv Shankar, Siddhant Garg, and Sunita Sarawagi. 2018. Surprisingly easy hard-attention for sequence to sequence learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 640–645.

Shikhar Sharma, Piyush Kumar, and Krishan Kumar. 2017. Lexer: Lexicon based emotion analyzer. In *International Conference on Pattern Recognition and Machine Intelligence*, pages 373–379. Springer.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

D Thenmozhi, B Senthil Kumar, and Chandrabose Aravindan. 2018. Ssn_nlp@ iecsil-fire-2018: Deep learning approach to named entity recognition and relation extraction for conversational systems in indian languages. *CEUR*, 2266:187–201.

Sayyed M Zahiri and Jinho D Choi. 2018. Emotion detection on tv show transcripts with sequence-based convolutional neural networks. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*.

SWAP at SemEval-2019 Task 3: Emotion detection in conversations through Tweets, CNN and LSTM deep neural networks

Marco Polignano
University of Bari A. MORO
Dept. Computer Science
E.Orabona 4, Italy
marco.polignano@uniba.it

Marco de Gemmis
University of Bari A. MORO
Dept. Computer Science
E.Orabona 4, Italy
marco.degemmis@uniba.it

Giovanni Semeraro
University of Bari A. MORO
Dept. Computer Science
E.Orabona 4, Italy
giovanni.semeraro@uniba.it

Abstract

Emotion detection from user-generated contents is growing in importance in the area of natural language processing. The approach we proposed for the EmoContext task is based on the combination of a CNN and an LSTM using a concatenation of word embeddings. A stack of convolutional neural networks (CNN) is used for capturing the hierarchical hidden relations among embedding features. Meanwhile, a long short-term memory network (LSTM) is used for capturing information shared among words of the sentence. Each conversation has been formalized as a list of word embeddings, in particular during experimental runs pre-trained Glove and Google word embeddings have been evaluated. Surface lexical features have been also considered, but they have been demonstrated to be not usefully for the classification in this specific task. The final system configuration achieved a micro F1 score of 0.7089. The python code of the system is fully available at `https://github.com/marcopoli/EmoContext2019`.

1 Introduction

The task of emotion detection from a text is growing in importance as a consequence of a large number of possible applications in personalized systems. This task can be considered as part of the sentiment analysis process also if it differs about the information collected. Sentiment Analysis aims to detect the polarity (positive, negative or neutral) about a topic of discussion or a specific aspect. On the contrary, Emotion Detection aims to associate an emotional label to textual content to explicitly understand what is the emotional state of the user while writing it. The final user behaviors are strongly influenced by the emotional state which she is in. Following the studies of Ekman (Ekman et al., 1987), Plutchik (Plutchik, 1990), Parrot (Parrott and Sabini, 1990), and Frijda (Frijda and Mesquita, 1994) some emotions can be considered "basics" and consequently more important than others during everyday decisions. Their identification is, therefore, one crucial aspect for applications in commerce, public health, disaster management, and trend analysis (consumer understanding). In the research area of emotion detection and sentiment analysis, many challenges are organized every for overcoming the state-of-the-art results. *SemEval* [1] is one of the most famous among them and it provides a large amount of data every year useful for supporting the research about the topic and commonly considered as state-of-the-art. Recently the best results are obtained by machine learning approaches (Colneriĉ and Demsar, 2018) based on recurrent neural networks (long short-term memory network) (Li and Qian, 2016; Wöllmer et al., 2010). These algorithms have quickly become the standard approach for solving the Emotion detection task placing great emphasis on the strategies used for formalizing the training data (Levy et al., 2015; Goldberg and Levy, 2014) and for optimizing hyper-parameters of the algorithms (Vilalta and Drissi, 2002).

2 Background and Related Work

Machine learning, and more recently deep learning algorithms, have been demonstrated to be the best option when approaching classification tasks of contents in natural language (Collobert and Weston, 2008). Example of state-of-the-art results have been achieved for hate speech detection (Zhang et al., 2018), part-of-speech tagging (Blevins et al., 2018) and name entity recognition (Chiu and Nichols, 2016; Chen et al., 2018).

Typical emotion detection systems work mostly with features directly extracted from text (Kao

[1] `http://alt.qcri.org/semeval2019/`

et al., 2009). A simple vector-space strategy can often be sufficient for resolving easier tasks, but it suffers from sparsity and lack of generalization. In (Bengio et al., 2003) the author exposes the concept of word embedding summarized as a ”learned distributed feature vector to represent similarity between words”. This concept has been exploited by Mikolov (Mikolov et al., 2013b) through word2vec, a tool for implementing work embeddings through two standard approaches: skip-gram (Guthrie et al., 2006) and CBOW (Mikolov et al., 2013a). An alternative word embedding representation is described in (Pennington et al., 2014) as Glove trained on global word-word co-occurrence counts and able to use statistics for producing a word vector space with meaningful sub-structure. However, the use of word embeddings enriched with surface lexical features is common in sentiment classification algorithms. The relevance of these features is supported by Mohammad et al. (Mohammad et al., 2013) that produced the top ranked system at SemEval-2013 and SemEval-2014 for sentiment classification of Tweets using emotional lexicons. Moreover, word and character n-grams, number of URL, mentions, hashtags, punctuations, word and document lengths, capitalization, and more are often used for improving the classification performances (Shojaee et al., 2013). A support for a correct classification is also provided by lexical resources used for look up the sentiment of words in sentences. Linguistic features include syntactic information such as Part of Speech (PoS) which can provide relevant information for formalizing the syntactical form of the sentence. These aspects have been considered in our final classification system in order to provide a robust and updated tool for emotion detection from Tweets.

3 The EmoContext task at SemEval 2019

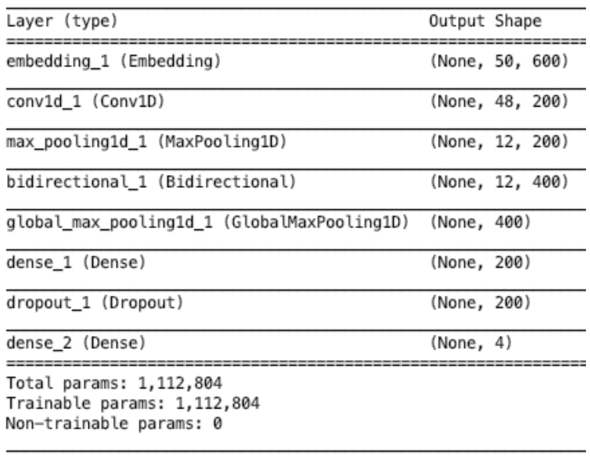
The EmoContext task at SemEval 2019 (Chatterjee et al., 2019) [2] aims to understand the emotion of the last turn expressed by a short dialog composed of three turns extracted from social media. The training set is composed of 30k records annotated with three main emotions: Happy, Sad, Angry and the ’other’ class that includes all other not annotated emotions following a data distribution of respectively 5k, 5k, 5k, 15k. The test set is composed by 5509 records, 2,95% of the total ’Happy’ , 2,68% about ’Sad’, and 3,15% of ’Angry’ records. The tuning of the systems has been performed over a ”dev set” composed by 2755 records with a class distribution similar to the one of the test set. Evaluation has been performed by calculating micro-averaged F1 score ($\mu F1$) for the three emotion classes, i.e. Happy, Sad and Angry.

[2] https://www.humanizing-ai.com/emocontext.html

4 Classification model

The model of emotion understanding applied in this study is based on the synergy between two deep learning classification approaches: the convolutional neural networks (LeCun et al., 1989) (CNN) and the long-short-term memory networks (LSTM) (Hochreiter and Schmidhuber, 1997).

The conjunct use of a CNN and an LSTM has been demonstrated to be very efficient with textual data (Chiu and Nichols, 2016; Ordóñez and Roggen, 2016). Fig. 1 shows the complete stack of the classification model for emotion understanding. Data are provided as the input of the model through a word embedding layer. Each n-gram of the record has been mapped into a k-dimensional word embedding vector. The dimension of the word embedding is different for each strategy of encoding evaluated, and the length of the record has been truncated at max 50 tokens. Words not found in the embedding dictionary have been encoded using a randomly selected word. The output of the previous layer has been provided to a 1D convolution layer with 200 filters and a kernel of size 3x3. The activation function used is the rectified linear unit function (’ReLU’) (Nair and Hinton, 2010). The output has been downsampled by a max pooling layer using a pool size of 4 along the number of tokens. The output of dimension 12x200 has been passed as input of a Bidirectional LSTM layer based, as for the CNN, on the ReLU activation function. The difference with a classic LSTM layer is the ability to find correlation among words in both the directions. In order to ’flatten’ the results, we used a max pooling strategy for considering only the highest value obtained for each slot and each direction. The resultant 1x400 vector has been provided to a dense layer without activation function with the purpose to reduce the dimensionality of the vector obtained. Finally, another dense layer with a soft-max activation function has been applied for estimating the probability distribution of each of the four classes of the dataset. The model has



```
Layer (type)                                  Output Shape
==============================================================
embedding_1 (Embedding)                       (None, 50, 600)
______________________________________________________________
conv1d_1 (Conv1D)                             (None, 48, 200)
______________________________________________________________
max_pooling1d_1 (MaxPooling1D)                (None, 12, 200)
______________________________________________________________
bidirectional_1 (Bidirectional)               (None, 12, 400)
______________________________________________________________
global_max_pooling1d_1 (GlobalMaxPooling1D)   (None, 400)
______________________________________________________________
dense_1 (Dense)                               (None, 200)
______________________________________________________________
dropout_1 (Dropout)                           (None, 200)
______________________________________________________________
dense_2 (Dense)                               (None, 4)
==============================================================
Total params: 1,112,804
Trainable params: 1,112,804
Non-trainable params: 0
______________________________________________________________
```

Figure 1: Model of emotion understanding using CNN and Bidirectional LSTM.

been trained using the categorical cross entropy loss function (Goodfellow et al., 2016) and Adam optimizer (Kingma and Ba, 2014).

5 Data processing

Each discussion in the dataset is provided as a set of three consecutive turns. We consider the dialog as a single textual content obtained concatenating the three turns into a single textual entity. Textual data have been processed for obtaining surface lexical features over the whole record. In particular, we calculate the following:

- **Statistics (RStat)**: number of tokens and characters; percent of uppercase characters and special tokens such as numbers, email, money, phone numbers, date and time, emoticons, stopwords, names, verbs, adverbs, pronoun; percent of punctuations including white spaces, exclamation points and word in a common words English dictionary [3];
- **Sentiment (RSent)**: the polarity of the record obtained through Stanford CoreNLP [4] and the percent of positive/negative words analyzed by TextBlob [5];

The textual record has been normalized before their transformation into word embeddings. We performed the correction of misspellings and the stripping of repeated characters using the Ekphrasis[6] python library. The record has been consequently tokenized using the TweetTokenizer of the ”nltk” suite [7] and when required for the word embedding lookup, they have been transformed into lower case. For each token we calculate, other extra features:

- **Statistics (TStat)**: percentage upper case characters; percentage repeated characters, before the text normalization;
- **Sentiment (TSent)**: sentiment of the token obtained using TextBlob;
- **Sentiment (TLex)**: part of speech; name entity label; is exclamation mark; is question mark; is a stopword; is in a dictionary of common English Words;

The transformation of each token in a word embedding has been performed using the following pre-trained resources:

- **Google word embeddings (GoEmb)**[8]: 300 dimensionality word2vec vectors, case sensitive, composed by a vocabulary of 3 millions words and phrases that they trained on roughly 100 billion words from a Google News dataset;
- **Glove (GLEmb)**:[9]: 300 dimensionality vectors, composed by a vocabulary of 2.2 millions words case sensitive trained on data crawled from generic web pages;
- **Sentiment140 positive (SentPosEmb) and negative (SentNegEmb)**: word embeddings created over the tweets annotated in the Sentiment140 dataset [10]. We used a word2vec skip-gram strategy over a window of 5 positions, 30 epochs and considering only words counted at least five times in the dataset. We produced two word embeddings (one for positive tweets and one for negative) of 100 dimensionality vectors each case sensitive;
- **Generic Tweets (GTEmb)**: word embeddings created over 1.1 million of generic tweets in English language. As previously, we used the skip-gram strategy over a window of 5 positions, 30 epochs min word count of 5 for obtained 300 dimensionality vectors case sensitive.

[3]https://github.com/cbaziotis/ekphrasis
[4]https://stanfordnlp.github.io/CoreNLP/
[5]https://textblob.readthedocs.io/en/dev/
[6]https://github.com/cbaziotis/ekphrasis
[7]https://www.nltk.org
[8]http://mccormickml.com/2016/04/12/googles-pretrained-word2vec-model-in-python/
[9]https://nlp.stanford.edu/projects/glove/
[10]https://www.kaggle.com/kazanova/sentiment140

	Dimensions	***Accuracy***	***Precision***	***Recall***	*μ **F1***
GoEmb	300	0.87005	0.63309	0.53441	0.57958
GoEmb + SentEmb	500	0.87150	0.73141	0.53415	0.61740
GoEmb + GTEmb	**600**	**0.91742**	0.71223	**0.71016**	**0.71119**
GeEmb + SentEmb + GTEmb	800	0.91070	0.72661	0.68397	0.70465
GLEmb	300	0.87005	0.69304	0.52260	0.59587
GLEmb + SentEmb	500	0.86787	0.69784	0.51871	0.59509
GLEmb + GTEmb	600	0.86896	**0.81055**	0.53228	0.64258
GLEmb + SentEmb + GTEmb	800	0.88094	0.77697	0.56055	0.65125

Table 1: Results obtained by different formalization of records through word emebeddings.

	Dim.	**Accuracy**	**Precision**	**Recall**	**μF1**	**diff. μ F1**
GoEmb + GTEmb	***600***	***0.91742***	***0.71223***	***0.71016***	***0.71119***	**-**
all_Lex_features	638	0.85562	0.76627	0.59110	0.66738	*-0.0438*
- RStat	617	0.89574	0.71411	0.66123	0.68665	*-0.0245*
- Rsent	632	0.86214	0.73456	0.61756	0.67099	*- 0.0401*
-TStat	636	0.85146	0.77134	0.56713	0.65365	*- 0.0573*
-TSent	636	0.85214	0.74840	0.59232	0.66131	*- 0.0498*
-TLex	631	0.86467	0.78254	0.58713	0.67089	*- 0.0402*

Table 2: Results obtained by different formalization of records through word emebeddings.

6 Experiments, discussion and results

We began to configure the proposed model pointing attention on the strategy to formalize records. We decided to train our model for 10 epochs for each run using a batches size equal to 64 on the *train* dataset and validating the model on the *dev* dataset. For each run, we vary the word embedding formalization. In Tab. 1 are shown the results that allow us to observe how the concatenation of Google pre-trained word embeddings (GoEmb) and the words embeddings obtained by general tweets (GTEmb) is the most promising for the classification task in term of micro F1. It is also important to note that the value of precision obtained by the concatenation of Glove pre-trained word embeddings (GLEmb) and the GTEmb set is the higher obtained but very unbalanced with the recall. This is a clear index of the instability of the model. The second step performed in this tuning phase has been the inclusion of surface lexical features about the records and every single token. In order to understand the influence of each set of lexical features on the final micro F1 score, we performed an ablation test. The results in Tab. 2 demonstrate that lexical features, in this specific classification task and dataset do not contribute positively to the final performances of the model. As a consequence of this observation, we decided to do not use them in our model.

Following the goal to make the model robust, we decided to train it for its final configuration also on data which comes from the dev set about the classes Happy, Sad and Angry. Then we trained the model again for 10 times on 100 epochs, with a batch size of 64 using GoEmb + GTEmb for data embeddings with a validation set of 20% of training data and an early stop when the micro F1 of the validation would overcome 0.75. We obtained three final models with micro F1 respectively of 0.7714, 0.8078 and 0.78163. We used these final models to classify the test set adopting a majority vote algorithm of the predictions. This strategy has allowed us to reach a final evaluation score of 0.7089 in the final task leader-board.

7 Conclusion

In this work, we proposed a robust emotion detection classifier based on the synergy of a CNN and an LSTM deep learning algorithm. The model has been evaluated with different data formalization and configurations for finding the one which better fits the data provided for the EmoContext task at SemEval-2019. Future work will include the evaluation of other model shapes and deep learning algorithms in order to increase the final performances of the system. The source code is available at `https://github.com/marcopoli/EmoContext2019`.

8 Acknowledgment

This research has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement N. 691071.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Terra Blevins, Omer Levy, and Luke Zettlemoyer. 2018. Deep rnns encode soft hierarchical syntax. *arXiv preprint arXiv:1805.04218*.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Guangyu Chen, Tao Liu, Deyuan Zhang, Bo Yu, and Baoxun Wang. 2018. Complex named entity recognition via deep multi-task learning from scratch. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 221–233. Springer.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Niko Colneriĉ and Janez Demsar. 2018. Emotion recognition on twitter: Comparative study and training a unison model. *IEEE Transactions on Affective Computing*.

Paul Ekman, Wallace V Friesen, Maureen O'Sullivan, Anthony Chan, Irene Diacoyanni-Tarlatzis, Karl Heider, Rainer Krause, William Ayhan LeCompte, Tom Pitcairn, Pio E Ricci-Bitti, et al. 1987. Universals and cultural differences in the judgments of facial expressions of emotion. *Journal of personality and social psychology*, 53(4):712.

Nico H Frijda and Batja Mesquita. 1994. The social roles and functions of emotions.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*, volume 1. MIT press Cambridge.

David Guthrie, Ben Allison, Wei Liu, Louise Guthrie, and Yorick Wilks. 2006. A closer look at skip-gram modelling. In *Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC-2006)*, pages 1–4.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Edward Chao-Chun Kao, Chun-Chieh Liu, Ting-Hao Yang, Chang-Tai Hsieh, and Von-Wun Soo. 2009. Towards text-based emotion detection a survey and possible improvements. In *Information Management and Engineering, 2009. ICIME'09. International Conference on*, pages 70–74. IEEE.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yann LeCun et al. 1989. Generalization and network design strategies. *Connectionism in perspective*, pages 143–155.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Dan Li and Jiang Qian. 2016. Text sentiment analysis based on long short-term memory. In *Computer Communication and the Internet (ICCCI), 2016 IEEE International Conference on*, pages 471–475. IEEE.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *arXiv preprint arXiv:1308.6242*.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Francisco Javier Ordóñez and Daniel Roggen. 2016. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115.

W Gerrod Parrott and John Sabini. 1990. Mood and memory under natural conditions: Evidence for mood incongruent recall. *Journal of personality and Social Psychology*, 59(2):321.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Robert Plutchik. 1990. Emotions and psychotherapy: A psychoevolutionary perspective. In *Emotion, psychopathology, and psychotherapy*, pages 3–41. Elsevier.

Somayeh Shojaee, Masrah Azrifah Azmi Murad, Azreen Bin Azman, Nurfadhlina Mohd Sharef, and Samaneh Nadali. 2013. Detecting deceptive reviews using lexical and syntactic features. In *Intelligent Systems Design and Applications (ISDA), 2013 13th International Conference on*, pages 53–58. IEEE.

Ricardo Vilalta and Youssef Drissi. 2002. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95.

Martin Wöllmer, Angeliki Metallinou, Florian Eyben, Björn Schuller, and Shrikanth Narayanan. 2010. Context-sensitive multimodal emotion recognition from speech and facial expression using bidirectional lstm modeling. In *Proc. INTERSPEECH 2010, Makuhari, Japan*, pages 2362–2365.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting hate speech on twitter using a convolution-gru based deep neural network. In *European Semantic Web Conference*, pages 745–760. Springer.

SymantoResearch at SemEval-2019 Task 3: Combined Neural Models for Emotion Classification in Human-Chatbot Conversations

Angelo Basile[*], Marc Franco-Salvador[*], Neha Pawar[*], Sanja Štajner[*],
Mara Chinea Rios, and Yassine Benajiba

Symanto Research, Nürnberg, Germany
{angelo.basile, marc.franco, neha.pawar, sanja.stajner,
mara.chinea, yassine.benajiba}@symanto.net

Abstract

In this paper, we present our participation to the EmoContext shared task on detecting emotions in English textual conversations between a human and a chatbot. We propose four neural systems and combine them to further improve the results. We show that our neural ensemble systems can successfully distinguish three emotions (SAD, HAPPY, and ANGRY), and separate them from the rest (OTHERS) in a highly-imbalanced scenario. Our best system achieved a 0.77 F_1-score and was ranked fourth out of 165 submissions.

1 Introduction

Detecting emotions in text is a key task in many scenarios, such as social listening, personalised marketing, customer caring, or in building emotionally intelligent chat-bots: in this last case, the task complexity increases, since a bot's response might influence the user's emotion.

The EmoContext shared task (Chatterjee et al., 2019) was posed as a sequence classification task. Given a set of three conversational turns (human–bot–human), the goal is to predict the emotion of the third turn. The label space contains the emotions SAD, ANGRY and HAPPY, and the label OTHERS denoting anything else (emotional or non-emotional), as illustrated in Table 1.

In this paper, we present our approaches to EmoContext shared task, and describe our best system in details. Additionally, we show that: (a) this task is very difficult even for humans (Section 2.2); (b) for this task, neural approaches outperform a strong non-neural baseline (Section 4); (c) an ensemble of neural systems with different architectures significantly outperforms the best neural model in isolation (Section 4).

[*] The first four authors have contributed equally to this work and are ordered alphabetically.

2 Data

The data released by the organisers consist of English user-chatbot interactions occurring in an Indian chat room. An overview of the dataset is provided in Table 2. It can be seen that the label distribution is highly imbalanced, and different for the training set than for the development (dev) and test sets (a 14:18:18:50 distribution for the training set, and a 5:5:5:85 distribution for the dev and test sets). To overcome this issue we tested three strategies: (1) down-sampling the dataset to its smallest class; (2) up-sampling the emotion-related labels with an in-house dataset; and (3) up-sampling by duplicating a random portion of the dataset. None of these solutions worked, and therefore, we trained our best models using the data provided by the organisers.

2.1 Preprocessing

The language of this corpus presents many of the features of micro-blogging language: large use of contractions (e.g. *I'm gonna bother*), elongations (e.g. *a vacation tooooooo!*), non-standard use of punctuation (e.g. *gonna explain you later..!*), incorrect spelling (e.g. *U r*).

To properly handle this language, we build a simple preprocessing pipeline which consists of: (1) the NLTK `TweetTokenizer` (Bird and Loper, 2004); and (2) a normalisation strategy that reduces sparseness by lowercasing all the words and converting elongations like *looool* to *lol*. These steps are used in all the experiments. Some of our models use additional preprocessing described further in the text.

ID	TURN 1	TURN 2	TURN 3	LABEL
71	Not good	:(why not..?	Been sick for one week	SAD
78	I hate Siri and it's friends	if you hate them , they are not your friends then xD	Yeah and u r Siri's friend so I hate utoo	ANGRY
91	Now I'm doing my dinner	I can see you!	How can you see me??	OTHERS
140	How about you tired of life or just your day?	Aha I' happy today, thanks for asking	Wow great..!	HAPPY

Table 1: Examples from the training dataset.

CLASS	TRAIN	DEV	TEST
SAD	5463	125	250
HAPPY	4243	142	284
ANGRY	5506	150	298
OTHERS	14948	2338	4677
total	30160	2755	5509

Table 2: Distribution of classes.

2.2 Manual Validation

To check how difficult this task is, for a trained human annotator, and get an estimate of the expected upper limit for our classification models, we asked two fluent (but non-native) English speakers with previous annotation experience to label 300 randomly selected instances from the dev set. The annotators achieved the official F_1-score of 0.73 and 0.72 against the 'gold' labels, and a 0.71 F_1-score among themselves. The only observed misclassifications between "emotional" classes were those between SAD and ANGRY. The highest number of disagreements the annotators had was between the OTHERS and the "emotional" classes. This showed that: (1) the task is naturally difficult (the trained human annotators reach 0.73 F_1-score at the most); (2) the main problem is distinguishing between the OTHERS class and the "emotional" classes.

3 Experimental Setup

We first randomly selected two times 2754 instances from the official training set, maintaining the class ratio that was announced for the official dev and test sets (4:4:4:88) resulting in 110 instances for the SAD, ANGRY and HAPPY, and 2424 instances for the OTHERS class. These two datasets we refer to as intDev and intTest sets, while the rest of the training dataset we refer to as intTrain.

We train and tune our four neural models (Section 3.1) using intTrain and intDev sets, and test them on the intTest, and the official dev and test sets (in different phases of the competition). We further experiment with combining their softmax output per class probabilities (Section 3.2).

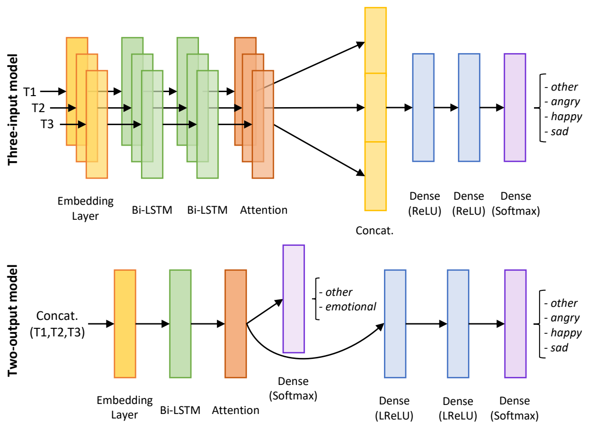


Figure 1: Model architectures for the three-input (IN3) and two-output (OUT2) models.

As a strong non-neural baseline we set up a linear SVM model with word and character n-grams (1-6) as features.[1]

3.1 Neural Models

We propose four neural network models that slightly differ on their objective.

3.1.1 Three-Input Model (IN3)

Having the three conversation turns (T1, T2, and T3), we explicitly represent the position of each sequence in the conversation by creating an input branch for each turn. The branches are identical and represent the text using word embeddings that feed a 2-layer bidirectional Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997). An attention mechanism (Yang et al., 2016) combines its hidden states. This architecture allows to independently process and attend to the most relevant parts of T1, T2, and T3. The information is later combined by a simple concatenation and few fully connected dense layers. The model architecture is shown in Figure 1.

We use a proprietary model © Symanto Research to obtain 300-dimensional word embeddings on the English Wikipedia. The performance

[1]For the implementation of the baseline we use `scikit-learn` (Pedregos et al., 2011). All the neural models are based on `Tensorflow` (Abadi et al., 2016), and for the ensemble models we use `Weka` (Hall et al., 2009).

of this representation is comparable with fastText (Bojanowski et al., 2017) but the resulting embedding model is fifty times lighter. We apply 10% dropout on the output of the embedding and concatenation layers, and layer normalisation (Ba et al., 2016) after the concatenation and before the output softmax.

3.1.2 Two-Output Model (OUT2)

Motivated by the findings of the manual validation (Section 2.2), we build this model in an attempt to ease the *emotional* vs. OTHERS classification. For this reason, we use a multi-task learning approach and add an auxiliary output whose label space conflated the ANGRY, HAPPY, and SAD labels into a single *emotional* one. We hypothesise that this approach is well suited to our unbalanced scenario, with the dominant OTHERS class.

The model architecture is similar to our three-input one (see Figure 1). However, the three conversational turns (T1, T2, and T3) are fed to the model as a single concatenated input, with additional tokens to mark the turn boundaries. The auxiliary output is connected to the output of the attention. This forces the attention weights to favour the *emotional* vs. OTHERS task.

We use the pretrained word embeddings described in Section 3.1.1. Our dense layers use the leaky version (LReLU) of the Rectified Linear Unit (ReLU) activation. In addition, we use the attention mechanism (He et al., 2017). Finally, we use the batch normalisation (Ioffe and Szegedy, 2015) to process the attention output.

3.1.3 Sentence-Encoder Model (USE)

As an exploration in transfer-learning, we build a simple feed-forward network together with a fine-tuned Universal Sentence Encoder (Cer et al., 2018). As input, we use the first (T1) and the last (T3) turn of the conversation, as we observed that adding the second turn (T2) leads to lower performances of this model.

3.1.4 BERT Model (BERT)

We fine-tune a BERT-base model (Devlin et al., 2018), modelling the problem as a sentence-pair classification problem: we use the first and the third conversational turn (T1 and T3) as the first and the second sentence respectively, completely ignoring the utterance by the bot (T2). We use this model in combination with a lexical normalisation system (van der Goot and van Noord, 2017).

We also built a neural model combining BERT, IN3, and OUT2, but it resulted in lower performance than any of those models separately, and is thus not presented here.

3.2 Ensemble Models

As we noticed that our neural systems have different strengths and weaknesses on the "emotional" classes (see Table 4), we combine them by using the softmax output probabilities of each class from all four models (16 features in total) and training several classification algorithms: Naïve Bayes (John and Langley, 1995), Logistic Regression (le Cessie and van Houwelingen, 1992), Support Vector Machines (Keerthi et al., 2001) with normalization (SVM-n) or standardization (SVM-s), JRip rule learner (Cohen, 1995), J48 (Quinlan, 1993), Random Forest (Breiman, 2001), and various meta-learners on top of them or their subsets.

The neural systems are trained and tuned on the intTrain and intDev sets, and their per class probabilities are obtained for the intTest, dev, and test sets. The ensemble models are then trained on the intTest+dev set and tested on the official test set. For this second classification stage, we thus have 5509 instances for training (intTest+dev) and 5509 for testing (the official test set).

4 Results

We evaluate our systems using precision (P) and recall (R) per each emotional class, and the micro F_1-score over the three "emotional" classes (the metric used by the task organisers for the official evaluation). The results for the baseline and the four neural systems are presented in Table 4. The results of the best ensemble models (trained on the per class probabilities of the four neural models) are presented in Table 5. We can notice that:

(1) Our best neural system (IN3) reaches .73 on the intTest set and .72 on the official test set.

(2) All our neural systems have a noticeably higher recall on the HAPPY and SAD classes on the intTest set than on the official dev and test sets.

(3) Our two best neural systems (IN3 and OUT2) have a noticeably lower precision on the HAPPY and SAD classes on the intTest set than on the official dev and test sets.

(4) Ensemble models reach .77 for three classification algorithms in the 10-fold cross-validation setup on the intTest+dev set, and that score is maintained on the official test set only by SVM.

ID	TURN 1	TURN 2	TURN 3	GOLD	OUR
388	Ok... No problem	ok i hope what you stay ok, be safe:)	Fuck off	OTHERS	ANGRY
4035	Am so pissed	one had been there for A MONTH	I'm so pissed	SAD	ANGRY
4129	yes. tomorrow :D	Yay, you.	hehehe gives sly smirk	OTHERS	HAPPY
397	What madness r u speaking abt?	what language do u think I'm speaking	English??	HAPPY	OTHERS
1640	You don't have it from outside	You have form the inside? Are you a sock?	??	ANGRY	OTHERS
253	wt u mean	I mean rest if the year	????	SAD	OTHERS

Table 3: Error analysis on the official test set.

SYSTEM	TEST	SAD		ANGRY		HAPPY		F
		P	R	P	R	P	R	
OUT2	int	.66	.90	.67	.87	.55	.84	.73
	dev	.75	.78	.65	.78	.62	.77	.72
	test	.71	.82	.64	.76	.65	.74	.71
IN3	int	.68	.92	.68	.84	.53	.88	.73
	dev	.71	.78	.67.	.81	.61	.76	.72
	test	.70	.78	.67	.81	.62	.76	.72
USE	int	.65	.84	.44	.93	.50	.89	.65
	dev	.68	.78	.47	.92	.56	.78	.66
	test	.68	.76	.48	.92	.58	.76	.66
BERT	int	.59	.92	.48	.92	.47	.88	.65
	dev	.61	.82	.51	.91	.50	.70	.64
	test	.59	.82	.54	.89	.51	.74	.65
baseline	int	.51	.89	.47	.85	.46	.81	.60
	dev	.57	.78	.47	.86	.54	.77	.63
	test	.55	.82	.48	.90	.52	.70	.63

Table 4: Results of our four neural systems and the strong non-neural baseline on the intTest (int), and the official development (dev) and test (test) sets.

SYSTEM	TEST	SAD		ANGRY		HAPPY		F
		P	R	P	R	P	R	
Logistic	CV	.81	.81	.76	.80	.72	.73	.77
	test	.83	.76	.74	.77	.77	.64	.75
SVMn	CV	.81	.83	.74	.80	.66	.76	.77
	test	.82	.80	.73	.79	.75	.72	**.77**
RanForest	CV	.82	.82	.81	.73	.73	.67	.76
	test	.83	.76	.77	.72	.80	.63	.75

Table 5: Results of our best ensemble models in a 10-fold cross-validation setup on intTest+dev (CV), and training on intTest+dev and testing on test set. Our best system submitted to the competition is marked in bold.

5 Error Analysis

The confusion matrix for our best system is given in Figure 2. The highest number of confusions is between the HAPPY and OTHERS classes, followed by confusions between the ANGRY and OTHERS.

Given the findings of our manual validation (Section 2.2), we performed an additional experiment. All instances for which our best system did not predict the gold label (355 instances), we presented to one of our annotators together with its gold and predicted labels (in random order), and asked him to choose the correct one, or assign a NOT SURE label. The annotator chose the label predicted by our system in 46% of the cases, the gold label in 39% of the cases, and in 15% of the cases the annotator was not sure. Several examples of instances for which the predicted label did not match the "gold" label are presented in Table 3.

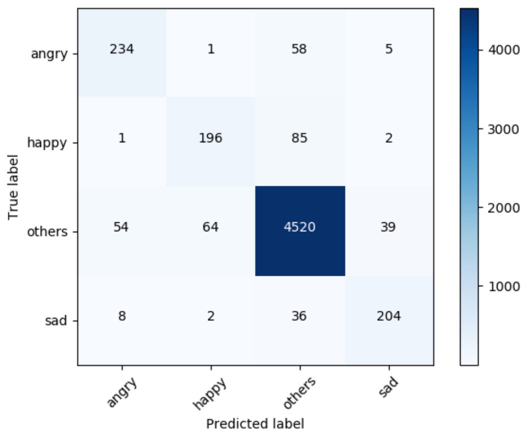


Figure 2: Confusion martix for the best model.

6 Conclusions

We presented our most successful approaches to the EmoContext shared task, with the goal of predicting the emotion (SAD, HAPPY, ANGRY, or OTHERS) in the third turn of a human–chatbot–human interaction, with an additional challenge of having a very unbalanced distribution of classes.

We showed that the task is difficult even for trained human annotators, and that our best neural systems can reach the human performance (.72 F-measure). Furthermore, we showed that a SVM classifier trained on the softmax output per class probabilities of four different neural systems can improve results scoring a .77 F_1-measure over the three emotional classes, and reaching thus the fourth place in the official competition.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Leo Breiman. 2001. Random Forests. *Machine Learning*, 45(1):5–32.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. *CoRR*, abs/1803.11175.

Saskia le Cessie and Johannes C. van Houwelingen. 1992. Ridge Estimators in Logistic Regression. *Applied Statistics*, 41(1):191–201.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

William W. Cohen. 1995. Fast Effective Rule Induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Rob van der Goot and Gertjan van Noord. 2017. Monoise: Modeling noise using a modular normalization system. *arXiv preprint arXiv:1710.03476*.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11:10–18.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017. An unsupervised neural attention model for aspect extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 388–397.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

George H. John and Pat Langley. 1995. Estimating Continuous Distributions in Bayesian Classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345.

S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. 2001. Improvements to Platt's SMO Algorithm for SVM Classifier Design. *Neural Computation*, 13(3):637–649.

Fabian Pedregos, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Appendix A: Model Parameters

Three-input model parameters: 20k most frequent tokens per branch, maximum text length of 25, 1024 LSTM units per layer, 300-dimensional dense layers, batch size of 128, 15 training epochs, and the Adam weight optimization.

Two-output model parameters: 20k most frequent tokens, maximum text length of 100, 300 LSTM units, batch size of 128, dense layer sizes of 300 and 150 (respectively), 10 training epochs, and the Adam weight optimization.

TDBot at SemEval-2019 Task 3: Context Aware Emotion Detection Using A Conditioned Classification Approach

Sourabh Maity
Teradata India Pvt. Ltd.
Hyderabad, India
sourabh.maity@teradata.com

Abstract

This paper presents the system developed to detect the contextual emotion (*SemEval19 Task 3* (Chatterjee et al., 2019)) from conversational dialogue. The system models the fact that emotion of a dialogue depends on the context of the conversation and not independent. It uses multiple layers in the deep learning model where each layer *bootstraps* with the context of what has already been said in the conversation.

1 Introduction

Over the years, we are getting more and more comfortable in text based conversations over the web, leading to increased interest in emotion analysis. Such a conversation is no longer limited between humans, it is now mainstream to use chat bots at various domains, e.g., customer care, HR management, virtual doctor etc.
Needless to say that in a conversation human emotions needed to be handled with care and empathy. Due to this, the task of emotion detection is very important when our aim to use chat bots and voice assistants more effectively. It is more difficult when the conversation is text based, lack of facial expressions and voice modulations make detecting emotions in text a challenging problem (Gupta et al., 2017).
In this SemEval19 Task 3 there were total three turns of dialogues; *turn1* and *turn3* were spoken by one participant of the conversation and *turn2* was spoken by another participant as a reply of *turn1*. We are tasked to detect the emotion of *turn3*. So, *turn1* and *turn2* act as context for *turn3*. There are four different emotions in the data set, namely, *happy*, *sad*, *angry* and *others*. The problem is modeled as a four class classification problem where each of the emotions listed above is the target class.

2 System Description

2.1 Preprocessing

This section describes the preprocessing steps of the system. Few of the steps are standard; the steps are just mentioned and are not discussed in detail. Rather the steps which are critical for the performance in the task are discussed in detail. Standard steps are: converting all letters to lower case, removing numbers, removing white spaces, removing stop words, sparse terms and particular words. The most important preprocessing steps are:
Expanding abbreviations: In chat data there are infinite number of possible abbreviations or shorthand uses, most of which are not standard. Those abbreviations can not be left in the data set as is, because there are no embedding for those. In my system , it is chosen to expand the top 10% of such abbreviations and others are ignored. For this, I created a map of abbreviation to expansion manually by inspecting the data set.
Few examples: lol $\rightarrow$ laugh out loud, ur $\rightarrow$ you are etc.
Handling emojis: Emojis are the single most important piece of information in chat data. In most of the cases it is a huge clue about the emotion of the party in conservation. I had two options to handle emojis, one, to use some kind of embedding (Eisner et al., 2016) for emojis; two, convert emojis into text and then use word embedding. I chose to convert emojis into text; partly because of the robust performance of the word embeddings and partly because of lack of a proven quality embedding for emojis. Also, this conversion made the *weight of evidence* feature (see section 2.2.2) more effective.
Examples: 😄 $\rightarrow$ beaming face with smiling eyes, ☹ $\rightarrow$ sad face etc.
But, a conversion scheme shown in the above examples leads to infiltration of words like *face*, *with*. To avoid this, I created a list of stop-words

and removed those from the expanded text. With this modification the above examples will look like:
😁 $\rightarrow$ beaming smiling eyes, ☹ $\rightarrow$ sad.

2.2 Features

There were mainly two features, word embedding and weight of evidence. Each word in the conversation is embedded into a 300 dimensional embedding space and for each turn the weight of evidence is computed. I intentionally refrained myself from using any sentence encoder like BERT (Devlin et al., 2018) or ELMo (Peters et al., 2018), as I wanted to explore the lower level embedding of words rather than using sentence embeddings as back boxes.

2.2.1 Word Embeddings

In the system, word embeddings are created as an average of three word vectors, GloVe (Pennington et al., 2014), FastText (Bojanowski et al., 2016) and Paragram (John Wieting and Livescu, 2015). I used 300 dimensional word embeddings. The embedding vocabulary could cover ∼85% of the data set vocabulary (unique words in the data set) which in turn covered ∼97% of the entire text of the data set.

2.2.2 Weight of Evidence

Weight of evidence (WOE) is a measure of how much the evidence supports or undermines a hypothesis. Here the intention is to weigh the evidence of each word in determining the emotion of the conversation. WOE is defined as:

$$WOE_{word,event} = \ln \frac{\frac{N^{non-event}_{word}}{N^{totalnon-event}_{word}}}{\frac{N^{event}_{word}}{N^{totalevent}_{word}}}$$

where,
$N^{non-event}_{word}$: number of other class records that has the word
$N^{totalnon-event}_{word}$: total number of other class records
N^{event}_{word}: number of records of the class that has the word
$N^{totalevent}_{word}$: total number of records of the class

Top 1000 most common words for each of the emotion classes were collected and then their WOE is computed for each of the four emotion classes. In the example below the words are represented by four dimensional vector, those dimensions belong to the four emotion classes.

$WOE_{word,event}$	Word	
	`smiling`	`sad`
$WOE_{word,happy}$	0.9	0.2
$WOE_{word,sad}$	0.1	0.8
$WOE_{word,angry}$	0.2	0.1
$WOE_{word,others}$	0.6	0.5

Table 1: $WOE_{word,event}$ for words.

For each turn I add up the WOE vectors of the words in that turn. So, each turn also has a WOE embedding of four dimensions. This embedding is fed into the model as an auxiliary feature. When emojis were converted into text, the WOE vector of the words explaining an important emoji reflected the emotion nicely. Also, when an emoji is used multiple times, its effect is multiplied into the WOE embedding of the turn. For example, "😁 😁 😁 " in a turn produces the below WOE embedding:

$WOE_{word,event}$	turn = 😁😁😁
$WOE_{smiling,happy}$	2.7
$WOE_{smiling,sad}$	0.3
$WOE_{smiling,angry}$	0.6
$WOE_{smiling,others}$	1.8

Table 2: WOE embedding for turn "😁 😁 😁 ".

Please note that "😁 😁 😁 " was first converted into text as: beaming smiling eyes beaming smiling eyes beaming smiling eyes. In the above table WOE vector for the word "smiling" is shown. Similar exercise can be done for other words.

2.3 Deep Learning Model

Given the turns of a conversation, the target emotion label can be modeled in different ways. One, model the target label based on *turn3* only. Two, Consider all the turns as one single input of text (may be separated by EOS tokens) and from this learn the target label. But, none of the options are truly context aware. Construction of my model is based on the idea that every turn in a conversation builds on top of the previous turn. Also this task is treated as a multi-class classification problem where each emotion is treated as individual classes.
At the core of the system are three bi-directional

(Schuster and Paliwal, 1997) Gated Recurrent Unit (GRU) (Cho et al., 2014) layers, one each for the three turns in the conversation. Second and third layer are *derived* from their immediate previous layer. This is achieved by using the hidden states of a turn GRU layer to initialize the subsequent turn's GRU layer. Hence, when turn three layer starts with the hidden state of turn two layer which has already summarized the *context* of the ongoing conversation, it is building on top of the existing context. I see it as each layer is conditioned on the what has already been conversed before it. Used model is depicted in Figure 2. Then the additional features, i.e., the WOE values were introduced into the model by concatenating with the intermediate latent representation of the conversation.

2.3.1 Gated Recurrent Unit: GRU

A GRU unit (in Figure 1) can be represented by the following equations:

$$\begin{aligned} z_t &= \sigma\big(x_t U^z + h_{t-1} W^z\big) \\ r_t &= \sigma\big(x_t U^r + h_{t-1} W^r\big) \\ \tilde{h}_t &= \tanh\big(x_t U^h + (r_t * h_{t-1}) W^h\big) \\ h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \end{aligned}$$

Here r is the reset gate, and z is the update gate. Intuitively, the reset gate determines how to combine the new input with the previous memory, and the update gate defines how much of the previous memory to keep. And h_t is the new hidden state.

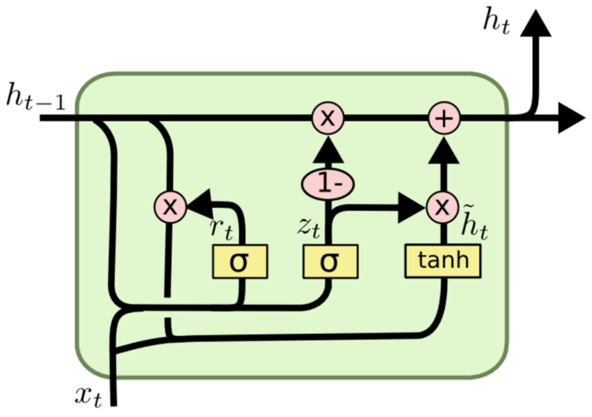


Figure 1: Gated Recurrent Unit. Figure adapted from (Olah, 2015).

2.3.2 Class Weights

The given data set is not well balanced (see Table 4 for details). To combat this issue I used $class\_weights$ for weighting the loss function, in a way it is to say the model which class to concentrate on. A *balanced* class weight is used to automatically adjust weights to be inversely proportional to class frequencies in the input training data. Weight of a class c_i is given by:

$$weight_{c_i} = \frac{n\_samples}{n\_class \times n\_samples_{c_i}}$$

Where,
$n\_samples$: total number of data sample
n_class: number of class present
$n\_samples_{c_i}$: number of samples of class c_i
The $class\_weights$ that were used are listed in Table 3.

class	$class\_weights$
$angry$	2.145
$happy$	0.841
sad	1.085
$others$	0.702

Table 3: $class\_weights$ for the input classes.

2.4 Data Description

We were provided 48544 data points to train our model. The class representations are shown in Table 4. It can be clearly seen that the data is highly imbalanced. This imbalance is handled by using weighted loss function and by fine tuning the model based on the micro-averaged f1 score (see section 2.5 for details).

Label	# data points
$angry$	5656
$happy$	14426
sad	11176
$others$	17286

Table 4: Class representation in training data.

2.5 Training Details

Data set is split (90 : 10) into train and validation. For class representation in the whole data set please see Table 4. In validation data generation, the proportion of class representation was kept similar to the data set. Table 5 shows the data split details.

I trained the model on the training data set and fine-tuned on the validation data set based on the

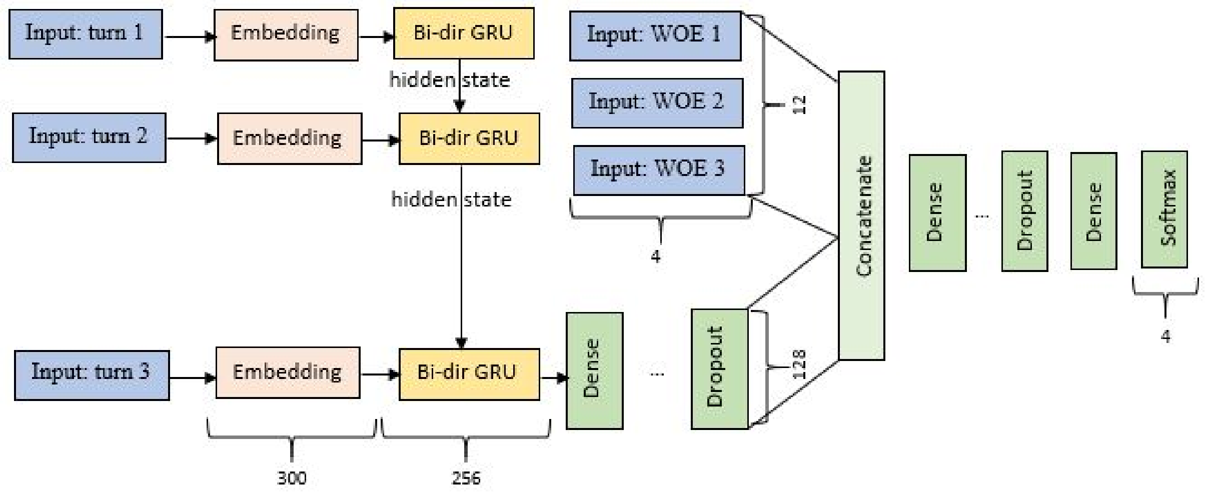


Figure 2: The deep learning model used in the system.

#Label	Training	Validation
#angry	5091	565
#happy	12983	1443
#sad	10058	1118
#others	15557	1729

Table 5: Class representation in training and validation data.

micro-F1 score. Since the data set is highly unbalanced, a weighted categorical cross-entropy loss is used, see Table 1 for the class weights. Adam (Kingma and Ba, 2015) optimizer is used with a learning rate of 0.001 and batch size of 128. Learning rate was decreased by 15% after each 3 epochs. Hidden state size of 256 is used for the bi-GRU gates. All the dense layers are of dimension 128 and a dropout of 0.5 is used for all of those.

3 Results

Here the detailed result of the system performance is presented. The performance shown in Table 6 is on the test data set.

label	precision	recall	f1-score	support
others	0.96	0.91	0.94	4677
happy	0.56	0.74	0.64	284
sad	0.61	0.80	0.69	250
angry	0.60	0.81	0.69	298

Table 6: System performance details

In the task the evaluation metric is micro-averaged F1 score only for the three emotion classes *happy*, *sad* and *angry*. Table 7 shows the confusion matrix of different classes.

label	*others*	*happy*	*sad*	*angry*
others	4246	162	119	150
happy	69	211	4	0
sad	36	3	200	11
angry	52	0	5	241

Table 7: Confusion matrix.

Precision and recall values for *happy*, *sad* and *angry* classes are 0.783653 and 0.589511 respectively. My system score is 0.6729 thereby beats the baseline (score 0.5868) convincingly.

4 Conclusion

With the system it is shown how to use the context information while detecting the emotion in a dialogue. Some guidelines about how to handle emojis is also laid out. While developing this system I realized the importance of pre-processing in conversational text data, or in general NLP related tasks; it can not be over emphasized.

Acknowledgments

I want to thank my mentors at Teradata, Ramesh Bhashyam and C Jaiprakash for the never ending support and to my teammates Lovlean Arora and Naveen TS for all the engaging discussions we had.
I want to apologize to my wife Samarpita, for all the weekend plans which were cancelled due to me working on TDBot. I promise to make up for those!

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR*, abs/1607.04606.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota, USA.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bosnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. *CoRR*, abs/1609.08359.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations. *CoRR*, abs/1707.06996.

Kevin Gimpel John Wieting, Mohit Bansal and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. volume 3, pages 345–358.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Christopher Olah. 2015. Understanding lstm networks. `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`. Online; visited 29/03/2019.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45:2673–2681.

THU_NGN at SemEval-2019 Task 3: Dialog Emotion Classification using Attentional LSTM-CNN

Suyu Ge , Tao Qi , Chuhan Wu , Yongfeng Huang
Department of Electronic Engineering, Tsinghua University Beijing 100084, China
{gesy17,qit16,wuch15,yfhuang}@mails.tsinghua.edu.cn

Abstract

With the development of the Internet, dialog systems are widely used in online platforms to provide personalized services for their users. It is important to understand the emotions through conversations to improve the quality of dialog systems. To facilitate the researches on dialog emotion recognition, the SemEval-2019 Task 3 named EmoContext is proposed. This task aims to classify the emotions of user utterance along with two short turns of dialogues into four categories. In this paper, we propose an attentional LSTM-CNN model to participate in this shared task. We use a combination of convolutional neural networks and long-short term neural networks to capture both local and long-distance contextual information in conversations. In addition, we apply attention mechanism to recognize and attend to important words within conversations. Besides, we propose to use ensemble strategies by combing the variants of our model with different pre-trained word embeddings via weighted voting. Our model achieved 0.7542 micro-F1 score in the final test data, ranking 15th out of 165 teams.

1 Introduction

The analysis of emotions in dialog systems where limited number of words appear with strong semantic relations between them deserves special attention in domain of natural language processing (NLP) due to both interesting language novelties and wide future prospects (Gupta et al., 2017). By analyzing the emotions through conversations, service providers can design better chatting strategies according to users' emotion patterns, which can improve user experience. Therefore, SemEval-2019 task 3 (Chatterjee et al., 2019) aims to call for research in this field. Given a textual dialogue, i.e., a user utterance along with two turns of context, systems need to classify the emotion of user utterance into four emotion classes: happy, sad, angry or others.

The field of sentiment analysis has been extensively studied. For example, SemEval-2018 Task 2 (Barbieri et al., 2018) once called for the study on relevance between tweet texts and emojis. However, understanding textual conversations is challenging in absence of voice modulations and facial expressions, which participants at this task are asked to deal with. Apart from diminishing the negative impact caused by class size imbalance, ambiguity, misspellings and slang, their systems should mainly focus on capturing the intricate interplay between two turns of conversations.

Traditional sentiment analysis requires a lot of feature engineering, such as n-grams and features extracted from sentiment lexicons (Mohammad and Turney, 2013; Kiritchenko et al., 2014a), and then feed them into a classifier such as Support Vector Machines (SVM) (Bollen et al., 2011; Kiritchenko et al., 2014b). However, manual feature engineering usually needs a large amount of domain knowledge. With the rapid development and ambiguity of social dialogues, these feature engineering strategies fade gradually and begin to be supplanted by neural networks (Tang et al., 2015; İrsoy and Cardie, 2014; Wang et al., 2016), which usually take word embeddings as inputs to incorporate rich semantic and syntactic information (Collobert and Weston, 2008). However, dialog emotion analysis is still very challenging, since dialog conversations can be very noisy and informal. In addition, the emotions evoked by conversations are usually highly context-dependent.

In this work, we propose an end-to-end attentional LSTM-CNN network as a unified model without hand-crafted features. In our approach, we use a combination of LSTM and CNN to capture both local and long-distance information. We use attention mechanism to select important words to learn more informative word representations. In addition, we use a data balancing method by setting a cost-sensitive loss function for training.

Besides, we use ensemble strategies by using a combination of the variants of our model with different pre-trained word embeddings. Our model achieved 0.7542 micro-F1 score on the test set, and extensive experiments validate the effectiveness of our approach. The source code can be found in our repository on github.[1]

2 Our Approach

The framework of our attentional LSTM-CNN model is illustrated in Figure 1. Each layer of network is introduced from bottom to top in the following sections.

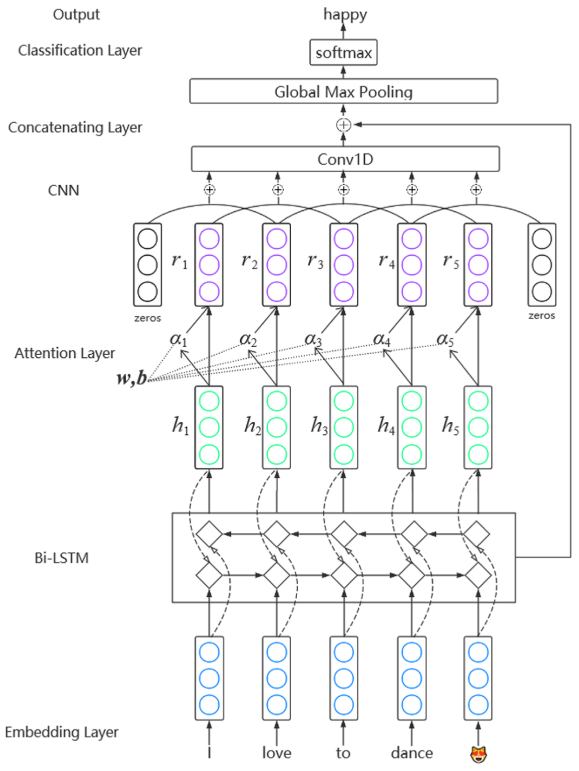


Figure 1: The architecture of our attentional LSTM-CNN model, the output is generated by soft voting ensemble after the softmax layer.

2.1 Word Embeddings

The first layer is a word embedding layer, which aims to convert the sequence of words in conversations into a low-dimensional vector sequence. We harness three types of pre-trained word embeddings, i.e., word2vec-twitter (Godin et al., 2015), pre-trained ekphrasis (Baziotis et al., 2017) vectors and GloVe (Pennington et al., 2014), to initialize the word embedding matrix.

[1]github.com/gesy17/Semeval2019-Task3-Emocontext

2.2 Bi-LSTM Layer

Considering the close relevance between two turns of dialogues, we use Bi-LSTM as encoder to capture abstract information from both directions. It consists of a forward LSTM $\overrightarrow{f}$ that encodes the sentence from $\boldsymbol{x_1}$ to $\boldsymbol{x_t}$ and a backward LSTM $\overleftarrow{f}$ that encodes the sentence backward. we concatenate the hidden representations in both directions, we get final representation of a word $\boldsymbol{x_i}$:

$$\boldsymbol{x_i} = \overleftarrow{\boldsymbol{x_i}} || \overrightarrow{\boldsymbol{x_i}} \quad \boldsymbol{x_i} \in \boldsymbol{R}^{2d}, \tag{1}$$

where $||$ denotes the concatenation operation and d is the size of each LSTM.

2.3 Attention Mechanism

An attention layer is incorporated after the Bi-LSTM layer to automatically select and attend to important words. The input of the attention layer is the hidden state vector $\boldsymbol{h_i}$ at each time step. The attention weight α_i for this time step can be computed as:

$$\boldsymbol{m_i} = \tanh \boldsymbol{h_i}, \tag{2}$$

$$\alpha_i = \boldsymbol{w}^T \boldsymbol{m_i} + b, \tag{3}$$

$$\alpha_i = \frac{\exp(\alpha_i)}{\sum_j exp(\alpha_j)}, \tag{4}$$

where w and b are the parameters of the attention layer. The output of attention layer at the i^{th} time step is:

$$\boldsymbol{r_i} = \alpha_i \boldsymbol{h_i} \tag{5}$$

2.4 CNN Layer

We use a convolutional neural network (CNN) to capture local contexts. Inspired by the residual connection for in ResNet (He et al., 2016), which combines the CNN outputs with original inputs to get better accuracy and shorter training time of deep CNN, we apply a merge layer to combine Bi-LSTM outputs and CNN outputs together. Our experiment proves that this structure can achieve a higher accuracy due to its full usage of both chronological information and local contextual information. Finally, max pooling is applied to the concatenated vectors to build conversation representations.

2.5 Emotion Classification

To make the final emotion prediction, we use a dense layer with softmax activation function to classify emotions. Considering the unbalanced data in both training set and testing set, we

	Happy	Sad	Angry	Average
Precision	0.7452	0.8117	0.7329	0.7598
Recall	0.6796	0.7760	0.7919	0.7488
F1	0.7109	0.7935	0.7613	0.7542

Table 1: Evaluation result on our final submission.

choose a cost-sensitive cross entropy loss function (Santos-Rodrguez et al., 2009) to modify the attention our model gives to different emotion categories. The loss function we use is formulated as:

$$\mathcal{L} = -\sum_{i=1}^{N} w_{y_i} y_i log(\hat{y}_i), \tag{6}$$

where N is the number of dialogue sentences, y_i is the emotion label of the i^{th} dialogue, $\hat{y}_i$ is the prediction score, and w_{y_i} is the loss weight of emotion label y_i. w_{y_i} is defined as $\frac{\sum_{k=1}^{C}\sqrt{N_k}}{\sqrt{N_{y_i}}}$, where C is the number of emotion categories and N_j is the number of texts with emotion label j. Consequently, this helps our model place higher weights towards infrequent emotion categories.

The last layer of our network utilizes a weighted soft voting ensemble method to fully take the advantage of different word embeddings. It should be mentioned that we design exactly the same network architecture with only word embeddings as slight differences. This soft voting method strengthens robustness and modifies our model to predict the class with the highest class probability.

3 Results and Analysis

3.1 Experimental settings

In our experiments, the word2vec-twitter embedding (Godin et al., 2015) was trained on 400 million microposts, which has a vocabulary of 3,039,345 words and 400-dimensional word representations. The Ekphrasis model leverages a collection of 330 million Twitter messages to generate word embeddings. It also uses GloVe as pretrained word vectors. Besides, a pre-processing pipeline is developed to enable users to get word vectors in a directly numerical form[2]. We also incorporate the GloVe embedding model and select the cased 300-dimension version[3] obtained by training on 2.2M data crawling from the Internet, containing 840B tokens in total.

[2]github.com/cbaziotis/ekphrasis
[3]nlp.stanford.edu/projects/glove

With word2vec-twitter embedding and GloVe embedding, we send raw texts to NLTK TweetTokenizer and randomly generate word vectors for all emojis and those out of vocabulary words appearing more than 3 times. Moreover, as to ekphrasis embedding, we use the pipeline provided by it. The pre-processing steps included in it are: Twitter-specic tokenization, spell correction, word normalization, word segmentation (for splitting hashtags) and word annotation.

In the experiment, we pertain the original dimension of the word embeddings and send them to a 400 dimension Bi-LSTM, adding to totally 800 dimension in LSTM layer. In the next CNN layer, the number of filters is 256, with filter length of 3. After each layer, we employ dropout with a drop rate of 0.2 to mitigate overfitting. Additionally, rmsprop (Tieleman and Hinton, 2012) is chosen as optimizer and Keras library (Chollet et al., 2015) is used for implementation.

3.2 Performance Evaluation

The final submission which scores micro $F1$ 75.42 is equipped with both the attention mechanism and weighted soft voting ensemble. The final result is shown in table 1, it suggests that our model performs relatively lower on happy emotion due to lack of training data and ambiguity. We evaluate parts of our network in the following paragraphs. The baseline we use is LSTM-CNN architecture(**LSTM-CNN**), baseline with concatenating layer is denoted as **LSTM-CNN+CL**. Upon this, attention mechanism is added, which is written as **LSTM-CNN+CL+AT**. Finally, a weighted soft voting is introduced, namely **LSTM-CNN+CL+AT+WE**. The result comparison is shown in table 2.

Concatenating Layer. By combining the outputs of Bi-LSTM and CNN layer, the model learns both local feature and long-term context, with the most obvious improvement in Word2vec-twitter, F1 score increasing from 0.7307 to 0.7483.

Attention Mechanism. Adding attention into network helps our network select those more essential words in the case of Ekphrasis and Glove word embeddings, but Word2vec-twitter witnesses a slight decline. This may be due to the randomness of out-of-vocabulary words and emoji word vectors. Overall speaking, attention benefits the study of word importance to some degree.

Weighted Soft Voting Ensemble. We place

	Word2vec-twitter	**Ekphrasis**	**GloVe**
LSTM-CNN	0.7307	0.7313	0.7429
LSTM-CNN+CL	**0.7483**	0.7355	0.7450
LSTM-CNN+CL+AT	0.7388	**0.7392**	**0.7460**
LSTM-CNN+CL+AT+WE		**0.7542**	

Table 2: Results on test data under various system framework.

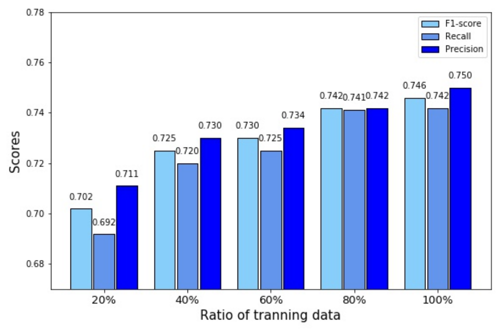


Figure 2: Influence of reducing training data on evaluation scores.

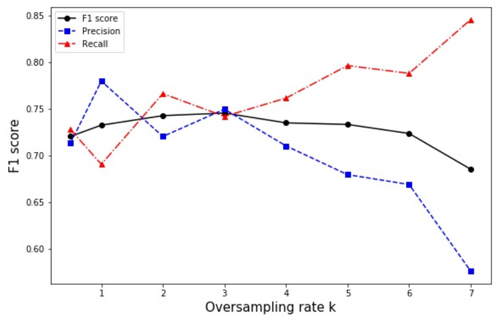


Figure 3: Influence of oversampling rate k on evaluation scores.

the highest weight on those showing good performances on dev dataset in our final submission. The significant improvement of F1 score at the bottom of table 2 indicates the power of ensemble. Results show that GloVe performs better than other two word embeddings, thus given more weight in practice. The result is ensembled from eight predictions , with the quantity we use of Word2vec-twitter, Ekphrasis, GloVe is respectively 2:3:3.

Quantity of training data. Since many methods in sentiment analysis rely heavily on high quality labeled data, we test our model with different reduction portion rate of training data. It can be seen in figure 2 that although there exists certain degree of performance reduction when the data amount is limited, our approach remain a F1 score of approximately 0.70 even with only 20% data, which proves that our approach can be widely applied to even when there exists shortage of labeled data.

Oversampling rate. The oversampling rate is defined to be the rate of loss weight between the class "others" and other three emotion categories. We officially set the oversampling rate k to be 3, meaning the loss weight rate between "others" and other three emotion categories is 3:1:1:1. To test the effectiveness of our choice of k, we select k to be in range from 0.5 to 7 and report the changes on F1 score, precision and recall in figure 3. It should be noticed that the scores are extremely unstable when $k < 2$, which may due to the sparsity of emotion labels in training data.

4 Conclusion

In this paper, we propose an attentional LSTM-CNN based neural network with concatenating layer for SemEval-2019 Task 3, i.e., predicting emotion categories of online dialogues. To strengthen robustness, weighted soft voting ensemble is exploited.

Acknowledgments

This work was supported by the National Key Research and Development Program of China under Grant number 2018YFC1604002, and the National Natural Science Foundation of China under Grant numbers U1836204, U1705261, U1636113, U1536201, and U1536207.

References

Francesco Barbieri, Jose Camacho-Collados, Francesco Ronzano, Luis Espinosa Anke, Miguel Ballesteros, Valerio Basile, Viviana Patti, and Horacio Saggion. 2018. Semeval 2018 task 2: Multilingual emoji prediction. In *Proceedings of*

The 12th International Workshop on Semantic Evaluation, pages 24–33. Association for Computational Linguistics.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada. Association for Computational Linguistics.

Johan Bollen, Huina Mao, and Alberto Pepe. 2011. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. *Icwsm*, 11:450–453.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Franois Chollet et al. 2015. Keras. `https://github.com/fchollet/keras`.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

Frderic Godin, Baptist Vandersmissen, Wesley De Neve, and Rik Van De Walle. 2015. Multimedia lab @ acl w-nut ner shared task: Named entity recognition for twitter microposts using distributed word representations. In *Workshop on User-generated Text*.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations. *arXiv preprint arXiv:1707.06996*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Ozan İrsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 720–728.

Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif Mohammad. 2014a. Nrc-canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 437–442. Association for Computational Linguistics.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. 2014b. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, 50:723–762.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a wordemotion association lexicon. *Computational Intelligence*, 29(3):436–465.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Ral Santos-Rodrguez, Dario Garca-Garca, and Jess Cid-Sueiro. 2009. Cost-sensitive classification based on bregman divergences for medical diagnosis.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2015. Target-dependent sentiment classification with long short term memory. *CoRR, abs/1512.01100*.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Yequan Wang, Minlie Huang, xiaoyan zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615. Association for Computational Linguistics.

THU-HCSI at SemEval-2019 Task 3: Hierarchical Ensemble Classification of Contextual Emotion in Conversation

Xihao Liang, Ye Ma, Mingxing Xu
Department of Computer Science & Technology
Tsinghua University, Beijing, China
`(liangxh16, ma-y17)@mails.tsinghua.edu.cn, xumx@tsinghua.edu.cn`

Abstract

In this paper, we describe our hierarchical ensemble system designed for the SemEval-2019 task3, *EmoContext*. In our system, three sets of classifiers are trained for different sub-targets and the predicted labels of these base classifiers are combined through three steps of voting to make the final prediction. Effective details for developing base classifiers are highlighted. Experiment results show that the ensemble approach manages to obtain better predictive performance in comparison with the base classifiers and our system has achieved the performance within the top 10 ranks in the final evaluation of *EmoContext*.

1 Introduction

Sentiment analysis is a task to identify the emotion conveyed by written language (Balahur et al., 2011; Lee et al., 2018). With the popularization of the Internet and instant message applications, text has become one of the most familiar media by which people express their ideas and communicate with each other. Automatic emotion classification can help people and robots better understand the messages or comments written by the others and make proper responses, which makes this study field increasingly important (Sun et al., 2018).

Deep learning approaches have achieved state-of-art performance in many recent studies of sentiment analysis (Rodrigues do Carmo et al., 2017). Gupta et al. (2017) used an LSTM-based model to identify the emotion in 3-turn conversations on Twitter. For emotion detection on TV show transcripts, a sequence-based convolutional neural network which can associate the information in the previous lines was designed by Zahiri and Choi (2017). Hazarika et al. (2018) proposed a conversational memory network based on both CNN (Lecun and Bengio, 1998) and gated recurrent unit (Chung et al., 2014) to recognize emotion in dyadic dialogue videos, featuring its ability to manipulate the information of different speakers. These studies focused on the architecture of the neural networks, but the optimization of the classification system based on the feature of the datasets are rarely discussed, which is crucial when dealing with specific real-world problems.

In this paper, we describe our approach to SemEval-2019 task3, *EmoContext* (Chatterjee et al., 2019), which aims to encourage more research of contextual emotion detection in textual conversation. Datasets of 3-turn conversations are provided and the participating systems are required to predict the contextual emotion of the last turn in the conversation: *Happy*, *Sad*, *Angry* or *Others*. The system performance is evaluated by the micro-averaged F1-score for *Angry*, *Happy*, and *Sad* (hereinafter referred to as *AHS*) on the given Test set. According to our observation of the dataset and single classifier's performance on it, we design a hierarchical ensemble classification system, which is composed of three sets of base classifiers, and three steps of voting to combine their predicted labels to make the final prediction. Our system has achieved F1-score of 0.7616 in the final evaluation, which is within the top 10 performances out of 165 participating systems.

This paper is organized as follows. In Section 2, our system and the strategies of training base classifiers are demonstrated. In Section 3, experiments are detailed and the evaluation results of our system and the base classifiers are presented and discussed. Conclusion is given in Section 4.

2 System Description

As we notice that the distinction among *AHS* and that between *Others* and *AHS* are significant, our system is designed to contain three sets of classifiers trained for different sub-targets and the pre-

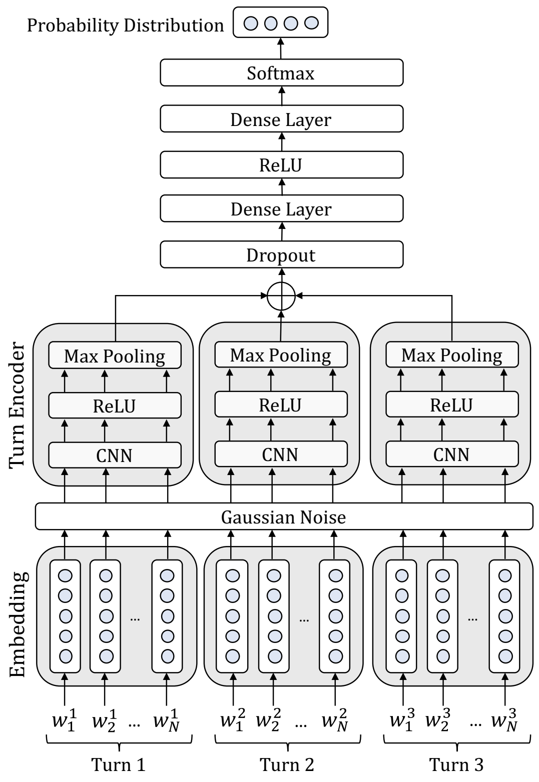


Figure 1: The architecture of the CNN-based classifiers in our system.

dicted emotion of a 3-turn conversation is obtained and refined through three steps of voting.

2.1 Classifier

Every base classifier in our system shares the same architecture (Fig 1). The input to the classifier is a 3-turn conversation represented as three sequences of tokens. An embedding layer is used to map the input to three sequences of vectors. Each turn is encoded to a vector individually by feeding its sequence of vectors into a CNN layer followed by a rectified linear unit (ReLU) and max pooling, as we notice that each turn's contribution to the prediction is different and the predictive clue to the contextual emotion can be captured within the turn. Three vectors are then obtained and concatenated as the feature vector of the 3-turn conversation. It is finally fed into a dense layer with a ReLU and another dense layer with softmax to get the probability distribution over all predicted classes.

2.2 Hierarchical Ensemble

Three steps of voting are designed (Algorithm 1) and a set of classifiers are trained for each step. For the first step, a set of four-emotion classifiers

Algorithm 1 Hierarchical ensemble

```
Input: the majority vote of Set A, $Label^A_{MV}$; the majority vote of Set B, $Label^B_{MV}$, and its voting count, $Count^B_{MV}$; the voting count for Others in Set C, $Count^C_{Others}$; two chosen thresholds, $thr_{II}$ and $thr_{III}$
Output: the predicted labels after each step of voting, $Label_I$, $Label_{II}$, $Label_{III}$;
 1: $Label_I \leftarrow Label^A_{MV}$;
 2: if $Label_I \neq Others$ & $Count^B_{MV} \geq thr_{II}$ then
 3:     $Label_{II} \leftarrow Label^A_{MV}$;
 4: else
 5:     $Label_{II} \leftarrow Label_I$;
 6: end if
 7: if $Count^C_{Others} >= thr_{III}$ then
 8:     $Label_{III} \leftarrow Others$;
 9: else
10:     $Label_{III} \leftarrow Label_{II}$;
11: end if
```

(hereinafter referred to as Set A) are trained on the given dataset and the original labels. For each test case, majority voting is applied to Set A to get the base predicted label, $Label^A_{MV}$. This set of predicted labels are hereinafter referred to as Prediction I.

For the second step, a set of three-emotion classifiers (hereinafter referred to as Set B) are trained only on the data of *AHS*. For each test case, majority voting is applied to Set B to get a new label $Label^B_{MV}$ and the predicted label for this case is changed to $Label^B_{MV}$ if $Label^A_{MV}$ is not *Others* and more than thr_{II} classifiers in Set B voted for $Label^B_{MV}$, by which the prediction for *AHS* is refined. This set of predicted labels are hereinafter referred to as Prediction II.

For the final step, a set of binary classifiers (hereinafter referred to as Set C) are trained on the given dataset but the labels for *AHS* are changed to *Not Others*. For each test case, we change its predicted label to *Others* if more than thr_{III} classifiers in Set C vote for *Others*, by which more *Others* samples are recalled. This set of predicted labels (hereinafter referred to as Prediction III) are used as the final prediction of our system.

2.3 Regularization

Common methods to alleviate the problem of overfitting are applied. Embedding layer is not fine-tuned. If this layer is tuned through training, the embedding space will be changed but the vectors of the tokens that exist only in the Test set are not adjusted, which may lead to wrong representations of these tokens. Gaussian noise is added after the embedding layer, by which the model is more robust when dealing with tokens of similar mean-

Turn 1	Turn 2	Turn 3	label
I live in uttra khand	ohh nice! love that place! ∧.∧	☺☺	happy
degreee	what degree & where?	sryyy i really got to goo	others

Table 1: Example training samples in Train set.

Dataset	Others	Happy	Sad	Angry
Train	14948	4243	5463	5506
Dev	2338	142	125	150
Test	4677	284	250	298

Table 2: Class distribution in each dataset.

ing. L2 regularization is applied to the weights of the stacked dense layers. Dropout (Srivastava et al., 2014) layer is added. Two more strategies are specifically highlighted.

- For each classifier, 90% of the training data is randomly selected for training and the 10% left for validation so that different information is captured. Although the evaluation metric is micro-averaged F1-score for *AHS*, for classifiers in Set A, the set of neural network weights that achieves the best micro-averaged precision on the validation set through the training process is chosen. As we notice that the precision is significantly sensitive to the class distribution and the task organizers have emphasized the difference between the class distributions of the provided datasets in advance, we need the precision of the base classifiers to be as high as possible.

- For each sample of the class *Others* in the training data, a new *Others* sample is created by randomly removing one token in one of the turns to simulate the situation when the user misspells a word and that misspelled token is not known to our embedding models. This is inspired by our observation that most of the *Others* samples in the training data still belong to *Others* even if one of the tokens is missing or misspelled. However, samples for *AHS* are not automatically generated in case a discriminative token is removed and a misleading sample will be made.

3 Experiments and Discussion

3.1 Data

Task organizers have released three datasets. Each sample in these datasets contains a 3-turn conversation(Table 1), in which Turn 1 was written by User 1, Turn 2 is User 2's reply to Turn 1 and Turn 3 is User 1's reply to Turn 2. The emotion label of Turn 3 is annotated for each conversation, which is one of the four emotions: *Angry*, *Happy*, *Sad*, and *Others*. Class distributions of these three datasets are shown in Table 2.

3.2 Preprocessing

We notice that the writing style of the conversations in the datasets resembles that of the tweets and comments on Twitter, featuring emoticons, informal usage of language, spelling errors and so on. Hence, we utilize the tweet processor, *ekphrasis*[1] (Baziotis et al., 2017). The preprocessing steps include (1) Twitter-specific tokenization, (2) spell correction, (3) word normalization for numbers and dates, (4) annotation for all-capital words, elongated words and repeated punctuations, (5) conversion of emoticons to emotion labels, through which each 3-turn conversation is converted to three sequences of tokens.

3.3 Word Embeddings

We deploy the pretrained embedding model provided by Baziotis et al. (2018) [2] (hereinafter called NtuaW2V) in our system, which contains 300-dimensional vectors trained on Twitter messages that are also preprocessed by *ekphrasis*. In addition, we tried another pretrained model provided by Mikolov et al. (2013) [3] (hereinafter called GoogleW2V), which contains 300-dimensional vectors trained on Google News dataset, in order to get an insight into the effect of different embedding models. The pretrained embeddings are used to initialize the embedding layer. For tokens that are not covered in the embedding model but occur in no less than two training samples, their embedding vectors are randomly initialized.

3.4 Implementation Details

Tensorflow (Abadi et al., 2016) is used to develop our models. For network optimization, we choose Adam algorithm (Kingma and Ba, 2014).

[1]https://github.com/cbaziotis/ekphrasis
[2]https://github.com/cbaziotis/ntua-slp-semeval2018
[3]https://code.google.com/archive/p/word2vec/

Parameter	Value
# of classifiers in each set	5
thr_{II}	2
thr_{III}	3
std. of Gaussian noise	0.1
kernel size of CNN	5
filter number of CNN	128
# of cells of the first dense layer	32
dropout keep prob.	0.5
l_2	0.2
initial learning rate	0.005
decay of learning rate	0.9
minibatch size	100

Table 3: Configuration of our system.

The configuration of our system and the hyper-parameters of the classifiers are shown in Table 3.

3.5 Results and Discussion

Table 4 shows the performance of our base classifier and its variants on the Test set. Note that **BASE** refers to the model trained as mentioned in Section 2 and **Prec**, **Rec**, and **F1** refer to the micro-averaged ones for *AHS*.

According to Table 4, adding automatically generated *Others* samples improves the accuracy, precision, and F1-score as more *Others* samples are correctly predicted but less *AHS* samples are recalled as the cost.

We also observe the performance difference when a different metric is used as the indicator for choosing the network weights, which is rarely discussed but the results show that the effect is significant. It is because of the remarkable difference between the class distributions of the training data and the Test set, which is emphasized by the task organizers.

On the other hand, the embedding models used to initialize the embedding layer are compared. Results show that using NtuaW2V achieves better F1-score while using GoogleW2V achieves the best precision but also the worst recall among the variants, which shows the importance of choosing suitable pretrained embedding models. The data on which the embedding model is trained and how the data are preprocessed should be the keys to it.

Table 5 and 6 illustrate the performance of our system after each step of voting. Results show that the first two steps bring slight improvement on all four metrics and the final step improves F1-score by raising the precision at the cost of the recall. The improvement also implies that, in comparison with Set A, classifiers in Set B are more effective to distinguish *AHS* samples and those in Set C are more precise to classify whether a sample belongs to *Others*. Although these classifier sets are all trained on the given dataset, Set B and Set C manage to work as a patch to Set A as they only focus on the simplified classification problems.

Classifier	Acc	Prec	Rec	F1
BASE	**0.9244**	0.7247	0.7661	**0.7445**
w/o extra Others	0.9206	0.6974	0.7932	0.7420
choose weight by				
F1-score	0.9112	0.6521	0.8093	0.7222
Recall	0.9020	0.6190	**0.8253**	0.7073
emb				
GoogleW2V	0.9226	**0.7421**	0.7163	0.7286

Table 4: Performance of the base classifier and its variants on the Test set.

Prediction	Acc	Prec	Rec	F1
I	0.9278	0.7360	0.7740	0.7545
II	0.9281	0.7383	**0.7764**	0.7569
III	**0.9305**	**0.7553**	0.7680	**0.7616**

Table 5: Performance of our system after each step of voting on the Test set.

4 Conclusion and Future Work

In this paper, we present our system used for SemEval2019 Task3, *EmoContext*. This system is composed of three sets of CNN-based neural network models trained for four-emotion classification, *Angry-Happy-Sad* classification and *Others*-or-not classification respectively. Three steps of voting are used to combine the predicted labels of the base classifiers and make the final prediction. Experiment results show the ensemble approach manages to improve the performance in comparison with the base classifiers. Automatic generation of random *Others* samples is proven effective and the importance of choosing pretrained embedding models and picking the right metric as the indicator for choosing network weights is highlighted. In order to achieve a better result based on our system, improving the performance of the base classifier is crucial. The architecture of neural networks and the features used as the input should be the fields that worth further exploration.

Prediction	Others	Happy	Sad	Angry
I	0.9595	0.7153	0.7806	0.7671
II	0.9595	**0.7190**	0.7801	0.7704
III	**0.9608**	0.7180	**0.7960**	**0.7709**

Table 6: F1-score for each emotion after each step of voting on the Test set.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, and Xiaoqiang Zheng. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems.

Alexandra Balahur, Jesús M Hermida, and Andrés Montoyo. 2011. Detecting implicit expressions of sentiment in text based on commonsense knowledge. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, pages 53–60. Association for Computational Linguistics.

Christos Baziotis, Nikos Athanasiou, Pinelopi Papalampidi, Athanasia Kolovou, and Alexandros Potamianos. 2018. Ntua-slp at semeval-2018 task 3: Tracking ironic tweets using ensembles of word and character level attentive rnns.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. pages 747–754.

Rodrigo Rodrigues do Carmo, Anísio Mendes Lacerda, and Daniel Hasan Dalip. 2017. A majority voting approach for sentiment analysis in short texts using topic models. In *Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web*, pages 449–455. ACM.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations.

Devamanyu Hazarika, Soujanya Poria, Amir Zadeh, Erik Cambria, Louis-Philippe Morency, and Roger Zimmermann. 2018. Conversational memory network for emotion recognition in dyadic dialogue videos. pages 2122–2132.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Yann Lecun and Yoshua Bengio. 1998. *Convolutional networks for images, speech, and time series*.

Gichag Lee, Jaey Un Jeong, Seungwan Seo, Czang Yeob Kim, and Pilsung Kan G. 2018. Sentiment classification with word localization based on weakly supervised learning with a convolutional neural network. *Knowledge-Based Systems*, 152:S0950705118301710.

Tomas Mikolov, Ilya Sutskever, Chen Kai, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26:3111–3119.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Xiaojie Sun, Menghao Du, Hua Shi, and Wenming Huang. 2018. Text sentiment polarity classification method based on word embedding. pages 99–104.

Sayyed M. Zahiri and Jinho D. Choi. 2017. Emotion detection on tv show transcripts with sequence-based convolutional neural networks.

TokyoTech_NLP at SemEval-2019 Task 3: Emotion-related Symbols in Emotion Detection

Zhishen Yang
Tokyo Institute of Technology
zhishen.yang@nlp.c.titech.ac.jp

Sam Vijlbrief
Delft University of Technology
s.vijlbrief@student.tudelft.nl

Naoaki Okazaki
Tokyo Institute of Technology
okazaki@c.titech.ac.jp

Abstract

This paper presents our contextual emotion detection system in approaching the SemEval-2019 shared task 3: *EmoContext: Contextual Emotion Detection in Text*. This system cooperates with an emotion detection neural network method (Poria et al., 2017), emoji2vec (Eisner et al., 2016) embedding, word2vec embedding (Mikolov et al., 2013), and our proposed emoticon and emoji preprocessing method. The experimental results demonstrate the usefulness of our emoticon and emoji prepossessing method, and representations of emoticons and emoji contribute model's emotion detection.

1 Introduction

Social media and online text applications have been gaining popularity in recent years. Users post videos, pictures, and text to share their daily life as well as to communicate with others. This vast amount of multimodal data greatly facilitates various user analysis tasks such as sentiment analysis.

Emotion detection as part of sentiment analysis can be conducted with user's multimodal data such as facial expression and voice data in addition to text data. Therefore, emotion detection becomes a challenging problem when only textual data is available for extracting the contextual and sentiment features (Chatterjee et al., 2019). For example, without proper visual and voice data, ”Why did you not call me last night” may be classified as sad or angry without an appropriate understanding of context.

The SemEval-2019 shared task 3: *EmoContext: Contextual Emotion Detection in Text* is the task to detect an emotion of a three-turn conversation (Chatterjee et al., 2019). We can consider this task as contextual sentiment analysis, as it requires detecting the emotion of the third-turn conversation by comprehensively understanding the contextual relationship and sentiment features from both language and emotion-related symbols.

Emotion class	Distribution
others	50%
happy	14%
sad	18%
angry	18%

Table 1: Emotion class distribution in the training data set.

The task organizers provided a training set that consisted of 30,160 three-turn conversations. Meanwhile, as Table 1 shows, the emotion class distribution in the training data set was unbalanced: 50% of samples were from “others” class. The task organizers also mentioned about the real-life distribution of emotion class: 88% of data is classified as “others” (Chatterjee et al., 2019). We also observed that over 28% of conversations in the training data set contained emoticons or emoji; users use them along with text to express emotion in a conversation. These two observations challenge us to have a method that can learn emotion and contextual features from unbalanced training data and the emotion-related symbols (emoticons and emoji).

The rest of this paper is organized as follows: Section 2 introduces the related work to the task; Section 3 explains our method in detail; Section 4 illustrates the experiments and analyzes the experimental results; Section 5 concludes the paper and presents the future work.

2 Related Work

Natural language processing in social media as an emergent area has attracted a lot of attention (Poria et al., 2017), especially from the recent advances in applying neural network methods with pre-trained embeddings (Eisner et al., 2016).

To achieve generalization and robustness in social media sentiment analysis, pre-trained embeddings should contain the representations of not only words from natural language but also emotion-related symbols, such as emoticons and emoji (Eisner et al., 2016). Both pre-trained embeddings GloVe (Pennington et al., 2014) and word2vec (Mikolov et al., 2013) do not contain representations for emotion-related symbols, which restricts the performance of sentiment analysis in social media. Although pre-trained emoji2vec embedding contains Unicode emoji representation, not all emotion-related symbols are included, such as emoticons.

As emotion detection is a part of sentiment analysis, and the data from the task organizers contains emoticons and emoji for emotion expressions, we can utilize a neural network method with pre-trained embedding to solve this task. We also need to address the lack of representations of emotion-related symbols.

3 Method

We formalize the SemEval-2019 shared task 3 as an emotion classification problem. Our method performs as an emotion classifier that accepts a conversation containing three-turn textual utterances, and classifies the last utterance to one of four pre-defined emotion class (happy, sad, angry, and others).

Our method focuses on learning contextual relationships and extracting emotion features from three-turn conversations. Poria et al. (2017) presented an LSTM-based contextual sentiment analysis model: contextual LSTM network that captures inter-dependency and contextual relationship among utterances in a video. Their experimental results demonstrated that contextual features of utterances significantly boost the performance of sentiment analysis; therefore, we decided to use this model as the main component in our system.

Although pre-trained embeddings such as GloVe and word2vec are easy to access, both of them do not have representations for emoticons and emoji (Eisner et al., 2016). However, pre-trained emoji2vec embedding as a supplement to pre-trained Google news word2vec (Mikolov et al., 2013) contains 1,661 emoji symbols and is ready to be augmented in downstream natural language processing tasks for social media (Eisner et al., 2016). Thus, we decided to concatenate word2vec and emoji2vec in our method as word embedding.

3.1 Contextual LSTM Network

Since bi-directional contextual LSTM (bc-LSTM) performed the best in the experiments of Poria et al. (2017), we selected this variant of contextual LSTM as the main component.

bc-LSTM (Poria et al., 2017) consists of five layers: 1) embedding layer; 2) input layer; 3) LSTM layer; 4) dense layer; and 5) softmax layer. The embedding layer (shared across three utterances) converts utterances into distributed representations. The input layer is a shared bi-directional LSTM, accepting the output of each utterance from the embedding layer in a sequence. The LSTM layer is an uni-directional LSTM that uses concatenation of outputs of utterances from the input layer. The extracted contextual features from LSTM layer feed into a dense layer. Finally, the softmax layer predicts an output from the dense layer.

3.2 Emoticon and Emoji Pre-possessing

The emoticon and emoji pre-processing method removes sentiment ambiguity that emoji bring and solves the lack of representations of emoticons in pre-trained emoji2vec (Eisner et al., 2016) embedding.

Emoji Normalization Since emoji2vec does not learn context-dependent definitions of emoji (Eisner et al., 2016), a mixture and duplication of emoji within textual data in an utterance will add the complexity and ambiguity in an emotion expression.

We also noticed that appending emoji to the end of an utterance did not change its sentiment. Instead, this process splits an utterance into two parts: text part and emoji part, which guarantees the smooth emotion expression in each part.

Our emoji normalization reduces multiple instances of an emoji into one instance and append it to the end of its belonging utterance.

Emoticon to Emoji Mapping In addition to emoji, emoticons also play a vital role in expressing emotions. Thus, representations for emoticons are also important to our method.

Although emoji2vec does not contain a representation of an emoticon, an emoticon can be treated as a ”surface variation” of an emoji. Thus, we can use the same emoji2vec representation of

Emoticon	Emoji
:3	😊
D;	😩
:-#	🤐
>:/	😕
:-o	😮
:-D	😃
:-#	🤐
:/	😕
=/	😕
:-o	😮
;-)	😉
>_>	😒
:-þ	😛
:-)	😊
>:¥	😕

Figure 1: Emoticon to Emoji Dictionary.

an emoji only if the emoticon is associated with the emoji. We built a dictionary to map an emoticon to its corresponding emoji (Figure 1). Containing 150 common emoticons, this dictionary can be used to replace emoticons to emoji.

3.3 System Description

Figure 2 shows an overview of our system: TokyoTech_NLP contextual emotion detection system (TNCED); our system consists of two phases: training and test phases. Both phases use the same pre-processing method. In the test phase, we use the contextual LSTM network classifier from the training phase to predict the emotion class on the test data.

4 Experiments

4.1 Data

We used the training data provided by the task organizers to train our model. For evaluation, we used the SemEval 2019 task 3 test data and micro F1 score and F1 score as metrics.

4.2 Experiment Setup

We used Keras (Chollet et al., 2015) and the code from the Github repository of bc-LSTM[1] to implement our TNCED system with the following settings: 128 LSTM dimensions; adam (Kingma and Ba, 2014) as an optimizer; 0.003 learning rate, 0.2 dropout, and 75 epochs. We used pre-trained Google word2vec (Mikolov et al., 2013) and pre-trained emoji2vec (Eisner et al., 2016) as embeddings.

4.3 Experiment Design

To evaluate the performance of *emoji normalization* and *emoticon to emoji mapping*, we used them to create following data set:

Data Set 1: For both training and test data, we first conducted *emoticon to emoji mapping*, and then performed punctuation normalization to reduce the number of repeated and frequently-used punctuation '?', '!', ',', '.' into one. We put a white space around the punctuation. Then, we applied *emoji normalization*.

Data Set 2: In addition to Data Set 1, we applied punctuation normalization to two frequently-used punctuation '_' and ':'; we only conducted *emoji normalization* in this data set. This data set was used in our submitted system for SemEval 2019 task 3.

Data Set 3: We did not apply any pre-processing in both training and test data.

We used these three data sets to train and test our system (TNCED), and calculated micro F1 score on the three emotion classes (happy, sad and angry) and F1 scores of happy, sad and angry classes.

4.4 Experimental Result

Table 2 shows micro F1 score and F1 scores of happy, sad, and angry classes obtained from the TNCED systems trained with different settings. The system with setting 1 (TNCED+Data Set 1) had the highest micro F1 score (0.7004) among the settings; it also achieved the highest F1 scores (0.746 and 0.709) in both sad and angry classes.

Compared with setting 3 (without any pre-processing), both settings 1 and 2 gained better F1 scores of three emotion classes as well as micro F1 scores, especially the setting 1 improved nearly 0.131 in F1 score of sad class and 0.071 in

[1]https://github.com/SenticNet/conv-emotion/tree/master/bc-LSTM

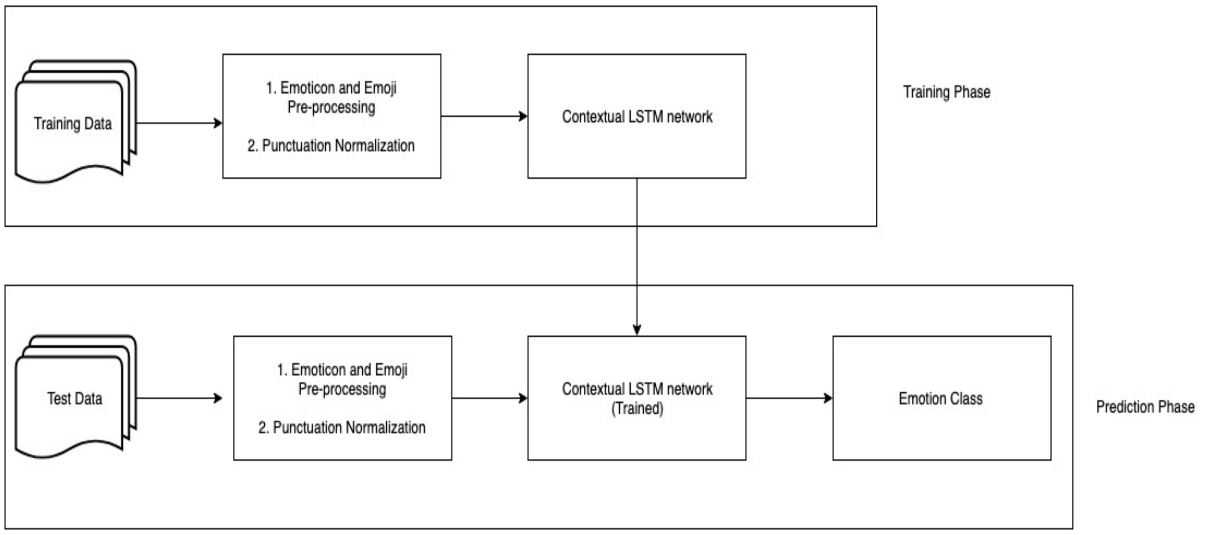

Training Data
1. Emoticon and Emoji Pre-processing
2. Punctuation Normalization
Contextual LSTM network
Training Phase
Test Data
1. Emoticon and Emoji Pre-processing
2. Punctuation Normalization
Contextual LSTM network (Trained)
Emotion Class
Prediction Phase

Figure 2: TokyoTech_NLP contextual emotion detection system (TNCED).

Experiment	**Micro F1**	**Happy F1**	**Sad F1**	**Angry F1**
Setting 1 (TNCED + Data Set 1)	0.7004	0.650	0.746	0.709
Setting 2 (TNCED + Data Set 2) (Submitted)	0.6801	0.658	0.695	0.688
Setting 3 (TNCED + Data Set 3)	0.6294	0.606	0.615	0.663
emocontext_organizers	0.5868			

Table 2: Micro F1 score and F1 scores of happy, sad, and angry obtained from TokyoTech_NLP contextual emotion detection system with three different settings, and SemEval-2019 Task 3 baseline's micro F1 score.

micro F1 score. Compare to setting 2, *emoticon to emoji mapping* helped setting 1, improving 0.051 and 0.021 in F1 scores of sad and angry classes. These results indicate that our proposed emoticon and emoji pre-processing method helps the contextual LSTM network in understanding emotion expressions of emoticons and emoji.

5 Conclusion

In this paper, we described our system for SemEval-2019 shared task 3. This system consisted of the pre-processing method for emoticons and emoji, and bi-directional contextual LSTM network (Poria et al., 2017).

The proposed emoticon and emoji pre-processing method solves the lack of representations of emotion-related symbols (emoticons) in pre-trained embeddings, and removes the ambiguity in emotion expressions of emotion-related symbols. Experimental results demonstrate the usefulness of the emoticon and emoji pre-processing method by improving the representations of emoticons and emoji. It helped the neural network model to capture emotion expressions from emotion-related symbols.

A future direction of this work includes training embeddings to gather contextual definitions of all emotion-related symbols. Furthermore, we would like to explore other neural network architectures as well as retrieve more data to capture the nuance emotion in text.

6 Acknowledgement

The research results have been achieved by "Research and Development of Deep Learning Technology for Advanced Multilingual Speech Translation", the Commissioned Research of National Institute of Information and Communications Technology (NICT), Japan.

References

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

François Chollet et al. 2015. Keras. https://keras.io.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. *arXiv preprint arXiv:1609.08359*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh, and Louis-Philippe Morency. 2017. Context-dependent sentiment analysis in user-generated videos. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 873–883.

UAIC at SemEval-2019 Task 3: Extracting Much from Little

Cristian Simionescu, Ingrid Stoleru, Diana Lucaci, Gheorghe Balan,
Iulian Bute, Adrian Iftene

Faculty of Computer Science, ”Alexandru Ioan Cuza” University of Iasi, Romania
cristian@nexusmedia.ro, ingridstoleru@gmail.com,
{diana.lucaci22, balangheorghe1997}@gmail.com,
iulian.bute@gmail.com, adiftene@info.uaic.ro

Abstract

In this paper, we present a system description for implementing a sentiment analysis agent capable of interpreting the state of an interlocutor engaged in short three message conversations. We present the results and observations of our work and which parts could be further improved in the future.

1 Introduction

It is hard to understand emotions in textual conversations in the absence of voice modulations and facial expressions (Gupta et al., 2017). In sentiment analysis task researchers work on different levels of sentiment analysis: document (when are considered single topics documents), sentence (when single sentences are classified as positive, negative or neutral), entity or aspect (which deal with finding the aspects in the text and then classifying in respective aspect) (Liu, 2012).

Similar to sentiment analysis at sentence level, in last years tweets from Twitter were analyzed and classified (Zhang and Liu, 2017), (Kumar and Sebastian, 2012), (Mukherjee and Bhattacharyya, 2013) and (Singh and Husain, 2014). In the beginning, a binary classification was used, which linked opinions or opinions only to two classes: positive or negative. In (Pak and Paroubek, 2010) the authors proposed a model for classifying tweets in goals, positive and negative feelings using a classifier based on multinomial Naive Bayes to use features such as N-grams and tags POS (part- of-Speech). In (Parikh and Movassate, 2009) the authors have implemented two models, one based on the Naive Bayes bigrams model and one using Maximum Entropy to classify tweets. (Go et al., 2009) proposed a solution for analysing feelings on Twitter using distant supervision, the training data were tweets with emoticons, which are regarded as noise data. They built several well performing models using Naive Bayes, MaxEnt, and SVM. (Barbosa and Feng, 2010) have modeled an automated method to classify tweets using space features including retweets, hashtags, links, punctuation mark amazement in combination with words and POS features polarity. (Luo et al., 2013) have brought to light the difficulties that they encounter when they want to classify tweets. Spam and a variety of languages on Twitter make the task of identifying opinions very difficult.

In SemEval 2019, in Task 3, EmoContext: Contextual Emotion Detection in Text (Chatterjee et al., 2019), the organizers ask participants to classify users messages in one of four classes: *Happy*, *Sad*, *Angry* or *Others*. These are given in the context of another two previous messages. The textual dialogue is composed of short messages that appear to be from a chat conversation. In such a context, the users express their thoughts and ideas in a compact way. In this paper, we describe how we created one classifier to detect the sentiment of short messages such as tweets.

2 Related Work

2.1 Word Embeddings

In order to incorporate the meaning of the words in a software system that processes natural language, distributed representations of words in a vector space are used to achieve better results by grouping similar words.

Previous work (Mikolov et al., 2013) introduces two architectures *CBOW* (Continuous Bag-of-Words) and *Skip-gram model* for learning word representations using neural networks. The later is more efficient for small training data, generating better representations for the infrequent words (Naili et al., 2017).

When choosing the best representations for a certain training dataset, one can either use pre-trained word embeddings that were built using large general corpora or train their own embeddings on a specific corpus which is similar to the type of data the model will be working with. The advantage of the first approach is that the representations only need to add without any additional computational cost, meanwhile, the second one requires a large enough corpus that can lead to meaningful representations that can capture both syntactic and semantic regularities. While vectors like Word2Vec, GloVe (Global Vectors for Word Representation) or fastText capture the most frequent semantic meaning of the words, training new representation on social media data can bring a number of advantages such as embedding the specific informal language that is used on these platforms and comprising numerous words that might not be very frequent in general corpora (Rotari et al., 2017) and (Flescan-Lovin-Arseni et al., 2017).

3 System Architecture

In this section we will present the systems developed for the EmoContext task.

3.1 Data Pipeline

Starting off, a critical characteristic in our architecture was the ease of configuration of different parameters of our system. We want our system to require little additional work and troubleshooting when changing, adding or removing pre-processing, feature extractions or post-processing techniques.

As seen in Figure 1, the system can take any configuration and order of pre-processing, feature extraction and post-process methods as well as a model to be fed the data.

Since a lot of small changes would sometimes occur on the later stages of the pipeline, we implemented an auto-save feature in all components of the system which will simply use the cached processed data up to the point of the last modified step in the pipeline.

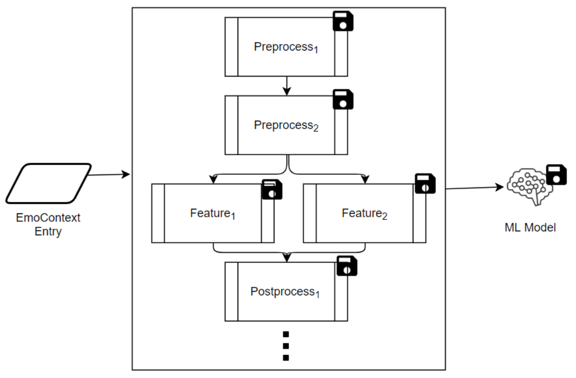


Figure 1: Pipeline structure

3.2 Data Processing

The dataset in from EmoContext task presented some clear challenges. Since we had to learn the expected sentiment of one of the interlocutors from a relatively small amount of text it was of utmost importance to remove noise from the data with minimal loss of potentially useful information. With such small amount of data (≈ 4 words per message) in each column, we decided to concatenate all three messages in every entry in order to be able to infer more information from it.

3.3 Pre-processing Stage

In the pre-processing stage, we implemented a number of steps progressively remove noise such as non utf-8 encoded symbols or random punctuation or characters, from which no important information could be extracted, using regex rewriting rules. After which we transform all misspelled words to the closest correct English word (closest in terms of Hamming distance) while some words would be transformed wrongly, the system performed better when using the "corrected" dataset.

We considered emoticons to be important in determining the sentiment state of the communicating parties, as such we identified as many standard use emoticons. With these emoticons, we analyzed the distribution of where they appear. For example: ":)" appeared predominantly in entries labeled "happy". Using these we replaced each emoticon with the keywords: "happyemoticon", "angryemoticon", "angryemoticon" and "otheremoticon" respectively.

In terms of the actual noise reducing rewriting rules:

- Eliminating any elongated series of characters greater than two, to a size of two;
- Reducing any repetition of English symbols or punctuation marks to just one, since it would not lead to any loss of information but it will remove noise;
- Removal of spaces between punctuation marks;
- Removal of any number with more than one decimal;
- Rewriting Unicode characters into utf-8 equivalents or complete removals when not possible;
- Isolated characters get deleted;
- Deletion of ASCII emoticons.

We have observed that this combination of preprocesses leads to the best results, without eliminating too much or leaving too much noise. Extensive empirical experimentation was done to assert the performance of various combinations of preprocesses and parameters.

We have to keep in mind, that before applying these modifications the average length of the concatenated messages is around 12 words, afterward, it became around 10.

3.4 Feature Selection

For the actual features we wanted to use in our system, we have attempted a number of syntactic features we thought of extracting.

All of these features proved to be either not helpful in the aid of the model performance or detrimental in the sense that it left any model we attempted prone to overfitting, such as getting stuck in the local maxima of classifying all instances as ”others”.

As such, we chose to use embeddings as our only form of feature selection. We have tried to utilize pre-trained embeddings offered online such as GloVe, FastText and Word2vec.

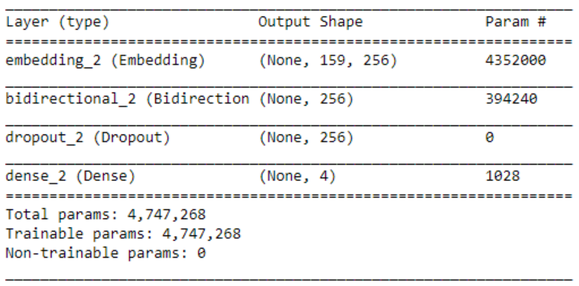
Sadly, all of the embeddings we have attempted to incorporate into the system produced weaker results compared to training the embedding from scratch on the data.

For this, we tokenized the data and padded it to have 200 elements per list. Even though these vectors were trained on a relatively small corpus, due to the high usage of jargon, rare abbreviations and bad grammar which made our dataset very much different compared to the corpus used by any of the above mentioned pre-trained word embeddings this was most likely the cause of improved performance when our own embedding even though the corpus is extremely small compared to what would be required to create a good embedding.

3.5 Model

In constructing our machine learning model, we chose to use artificial neural networks with the use of the ”Keras” python library[1].

For the actual model of the system, we have made use of very simple and small architectures since any attempt of creating a deeper or wider artificial neural network models resulted in drastic overfitting. Even with other overfitting alleviating techniques such as regularization, dropout and batch normalization we had to stick to a shallow architecture. We suspect this is due to the fact that we trained our embedding on such a small dataset, perhaps if more similar data can be collected and a more general word vector is created, overfitting would also be reduced. As such, the model we are presenting does not suffer from overfitting but it is relatively shallow.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_2 (Embedding)      (None, 159, 256)          4352000   
_________________________________________________________________
bidirectional_2 (Bidirection (None, 256)               394240    
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 1028      
=================================================================
Total params: 4,747,268
Trainable params: 4,747,268
Non-trainable params: 0
_________________________________________________________________
```

Figure 2: Model

As seen in Figure 2, we used a trainable embedding layer of size 256 as input to fit on our training data. Next we used a single hidden layer of 128 Bidirectional LSTM cells with a 30% dropout and a tanh activation function. Finally outputting the result in a 4 neuron layer using the softmax function to learn the correct expected labels.

The model was trained using the RMSprop optimization algorithm.

	Metrics				
	micro F1	Accuracy	Precision	Recall	Sensitivity
Others	**0.92459089**	0.87620258	**0.95740783**	**0.89394911**	0.77644231
Angry	0.61855670	0.94626974	0.50209205	0.80536913	0.95432738
Sad	0.63508772	**0.96224360**	0.56562500	0.72400000	0.97356912
Happy	0.56877323	0.95788709	0.60236220	0.53873239	**0.98066986**
Average	*0.68675210*	*0.93565070*	*0.65687170*	*0.74051260*	*0.92125210*

Table 1: Submission metrics

4 Results

Using that simple model and an extensive meta-parameter tuning we were able to reach an average micro F1 score of 0.6895, this being the last submission we were able to upload during the workshop. See Table 1 for complete metrics of this submission, calculated by training the model 5 times, to factor in for randomness of shuffled data and weight initialization (all runs had comparably similar results).

We noticed that the greatest difficulty our system faces is correctly classifying instances belonging to the ”happy” class. As such, we should look into what data pre-processing we could use in order to decrease the high number of false-negatives.

Another potential improvement would be to add weights to the loss function based on the proficiency we observe the system to exhibit on each type of entry.

Applying the pre-processing we described previously we managed to boost that result to average micro F1 of 0.7362.

Both of these models were trained using a K-fold cross-validation with four splits and a batch size of 64.

What we observed time and time again, the main issue we faced was overfitting of the training data, as we can see when looking at the progression on the validation data, see Figure 3.

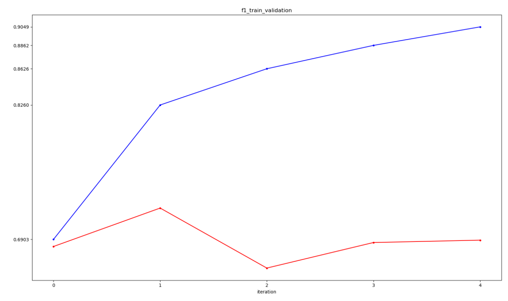


Figure 3: Training / Validation F1 - red validation data set, blue train data set

5 Conclusions

This paper presents the system developed by our group for the EmoContext task. The architecture of the system includes data processing, feature selection, and machine learning model. The results are promising, but they also expose the need for more experiments that should be done in this field in the next period.

For the future, we believe a CNN approach could prove fruitful. As well as a different or deeper network and configuration while using a similar pre-processing process which we believe is the main contributor to our relatively successful result.

Another direction worth investigating would be a Mixture of Experts approach, using various variations of the system even if they prove sub-optimal individually, such as *The currently proposed system*; *A system using a pre-trained embedding*; *Three sub-systems each trained to only classify one of the classes*; *A system with three input layers, one for each message reply, removing the concatenation of the text pre-processing*; *A system which only looks at the emoticons present in the text.*

Acknowledgments

This work is partially supported by POC-A1-A1.2.3-G-2015 program, as part of the PrivateSky project (P_40_371/13/01.09.2016).

References

Luciano Barbosa and Junlan Feng. 2010. Robust sentiment detection on twitter from biased and noisy

[1]https://keras.io/

data. volume 2, pages 36–44.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Iuliana Alexandra Flescan-Lovin-Arseni, Ramona Andreea Turcu, Cristina Sirbu, Larisa Alexa, Sandra Maria Amarandei, Nichita Herciu, Constantin Scutaru, Diana Trandabat, and Adrian Iftene. 2017. warteam at semeval-2017 task 6: Using neural networks for discovering humorous tweets. *SemEval@ACL 2017*, pages 407–410.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *Processing*, pages 1–6.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations. *CoRR*, abs/1707.06996.

Akshi Kumar and Teeja Mary Sebastian. 2012. Sentiment analysis on twitter. *IJCSI International Journal of Computer Science Issues*, 9.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Morgan Claypool Publishers.

Tiejian Luo, Su Chen, Guandong Xu, and Jia Zhou. 2013. *Sentiment Analysis*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *ICLR Workshop*.

Subhabrata Mukherjee and Pushpak Bhattacharyya. 2013. Sentiment analysis : A literature survey.

Marwa Naili, Anja Habacha Chaibi, and Henda Hajjami Ben Ghezala. 2017. Comparative study of word embedding methods in topic segmentation. *International Conference on Knowledge Based and Intelligent Information and Engineering Systems*.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*.

Ravi Parikh and Matin Movassate. 2009. Sentiment analysis of user-generated twitter updates using various classication techniques.

Razvan-Gabriel Rotari, Ionut Hulub, Stefan Oprea, Mihaela Plamad-Onofrei, Alina Beatrice Lorent, Raluca Preisler, Adrian Iftene, and Diana Trandabat. 2017. Wild devs at semeval-2017 task 2: Using neural networks to discover word similarity. *SemEval@ACL 2017*, pages 267–270.

Pravesh Kumar Singh and Mohd Shahid Husain. 2014. Methodological study of opinion minng and sentiment analysis techniques. *IJSC International Journal of Soft Computing*, 5.

Lei Zhang and Bing Liu. 2017. *Sentiment Analysis and Opinion Mining*. Springer US, Boston, MA.

YUN-HPCC at SemEval-2019 Task 3: Multi-Step Ensemble Neural Network for Sentiment Analysis in Textual Conversation

Dawei Li, Jin Wang and **Xuejie Zhang**
School of Information Science and Engineering
Yunnan University
Kunming, P.R. China
Contact: xjzhang@ynu.edu.cn

Abstract

This paper describes our approach to the emotion detection of Twitter textual conversations based on deep learning. We analyze the syntax, abbreviations, and informal-writing of Twitter; and perform perfect data preprocessing on the data to convert them to normative text. We apply a multi-step ensemble strategy to solve the problem of extremely unbalanced data in the training set. This is achieved by taking the GloVe and ELMo word vectors as input into a combination model with four different deep neural networks. The experimental results from the development dataset demonstrate that the proposed model exhibits a strong generalization ability. For evaluation on the test dataset, we integrated the results using the stacking ensemble learning approach and achieved competitive results. According to the final official review, the results of our model achieved micro-F_1 score of about 0.7588 on the final evaluation.

1 Introduction

Over the past 10 years, short microblogging communication methods, such as Tweets and Weibo, have been widely adopted. Numerous emotions exist in the dialogue of texts, and there is great commercial value for the detection of such emotions. For example, in the customer service field, the feedback time limit is divided according to the emotion.

Text conversation emotion detection is a challenging issue without facial expressions and mood information being available. Moreover, the data of sadness, anger, and happiness in the current context, as well as the extremely unbalanced scale, natural language ambiguity, and rapidly growing online language, further exacerbate the challenges of sentiment detection.

In this paper, we describe our work on SemEval 2019 Task 3, EmoContext: Contextual Emotion Detection in Text (Chatterjee et al., 2019). The main challenge of the task lies in imbalanced data distribution. Previously proposed methods for solving data imbalance include over-sampling, under-sampling (Weiss, 2004), and Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al., 2002). over-sampling is copying from a smaller number of categories, which may lead to over-fitting. Under-sampling discards potentially useful information, which can degrade the performance of the classifier (Drummond et al., 2003). SMOTE is down-sampling first, and then integration. We propose a multi-step integration approach similar to SMOTE for this task to alleviate the data imbalance problem. In the first step, we use five-fold cross-validation to train five sub-neural networks with different data distributions. Thereafter, soft-voting is used for integrating the results from these sub-neural networks. Soft-voting integration is applied because it not only alleviates the problem of unstable prediction results caused by data imbalance, but also improves the effective prediction accuracy of the single model. The second step of stacking integration further enhances the global effective prediction accuracy. The experimental results indicate that the proposed model alleviates the problem of data imbalance and significantly improves the effective prediction accuracy.

The remainder of this paper is organized as follows. In section 2, we describe the system architecture. Section 3 explains the data processing and parameter tuning. The conclusions and future work are presented in Section 4.

2 System Architecture

The data with the label *others* comprise 49.56% of the training set. The model trained by the conventional method exhibits a poor generalization a-

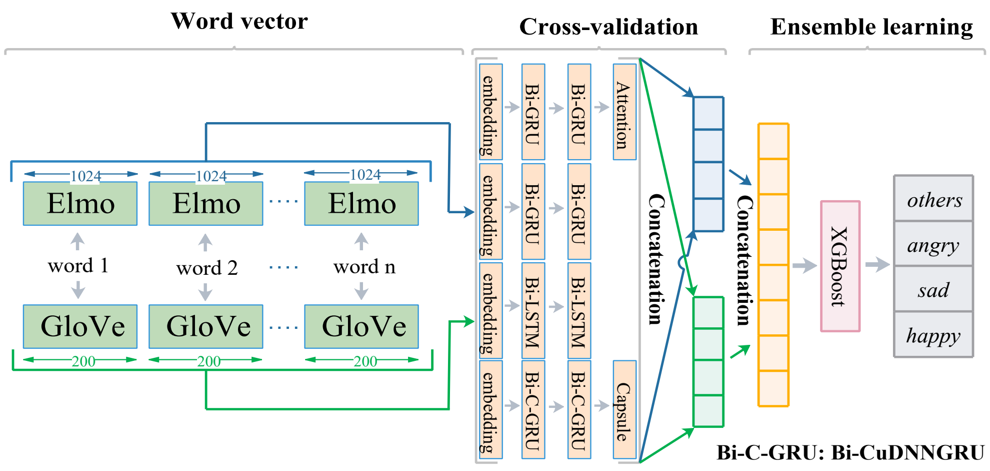


Figure 1: System architecture.

bility and is prone to over-fitting. Sampling verification may alleviate the problem of data imbalance (He and Garcia, 2008). In order to enable the model to learn the data characteristics of small samples, we use five-fold cross-validation to verify the model and test its robustness. The system architecture is illustrated in Figure 1.

2.1 Embedding

We use a GloVe (Pennington et al., 2014) pre-trained word vector: the Twitter 200-dimensional word vector. GloVe is a word representation tool based on the count base and overall statistics. It expresses a word as a vector of real numbers, and captures the semantic properties of words, such as similarity and analogy. Meanwhile, the ELMo algorithm (Peters et al., 2018) is used to train word vectors. ELMo simulates not only the complex features of vocabulary use, but also the changes in these usages in different language contexts. To train the ELMo word vector, we use the aforementioned processed text as input, including both the training and development set. The text is processed into a lookup table, and the words are passed into the ELMo algorithm one by one to generate a 1024-dimensional word vector.

For the feature extraction step, we use Keras (Francois and Chollet, 2015) to convert the text into a vector form of the word embeddings.

2.2 Model

Conventionally, the deep learning model is used in the natural language processing field. We use four superior-performance model components. By combining two different word embedding models, we obtain eight different models. We use four deep learning models, namely LSTM (Hochreiter and Schmidhuber, 1997; Mikolov, 2010), GRU (Cho et al., 2014), Capsule-Net (Sabour et al., 2017; Zhao et al., 2018), and Self-Attention (Luong et al., 2015).

We use Dropout (Salakhutdinov et al., 2014) to aid with improved model convergence. Finally, at each model output, we output the four predicted categories of probabilities, instead of the predicted results.

2.3 Ensemble Learning

According to the dataset characteristics, we design a multi-step ensemble neural network for the four-category emotion detection task.

The first step of integration consists of randomly dividing the entire dataset into 5-folds. In each round, four folds are used for training and the remaining fold is used to validate the model. Moreover, the model is used to predict the development and test sets. We use softmax activation function to get the probability distribution. At the end of the 5-fold cross-validation, five predicted probability values of train set, development set and test set are obtained. We apply a soft-voting mechanism to integrate the prediction probability on five predicted probability of development and test sets. The second step of integration involves combining different word vectors (GloVe and ELMo) with different models. The results of the first step of multiple models are horizontally concatenated as input for the second step of integration. The parameters are tuned and the predicted results are output.

Codes are publicly available at https://github.com/L-Maybe/SemEval-2019-task3-EmoContext

	others	*angry*	*sad*	*happy*
Training	0.4956	0.1826	0.1811	0.1407
Development	0.8486	0.0544	0.0454	0.0515

Table 1: Percentage of categories in dataset.

In the final integration phase, we use the XG-Boost (Chen and Guestrin, 2016) toolkit, which utilizes CPU multithreading for parallelism and exhibits strong classification performance. Furthermore, the toolkit can set different weights for unbalanced datasets, which is the most important reason for its use as the final predictive classifier. Following the ensemble learning of the soft-voting in the first step, we obtain eight groups of classification probability values. Owing to the different inputs, the final outputs of the four models differ, which meets the requirements of integrated learning. We horizontally concatenate the eight sets of probability prediction values into a new feature matrix and use the XGBoost tool to learn the new feature matrix to obtain the final prediction result.

3 Experiments and Results

3.1 Datastes and Official Evaluation Metrics

Datasets were provided by SemEval 2019 Task 3, EmoContext: Contextual Emotion Detection in Text (Chatterjee et al., 2019). The participants were asked to predict the emotions of a three-turn conversation. The task considered three emotion classes, namely *happy*, *sad*, and *angry*, along with an *others* category. The number of training and development sets was 30160 and 2755, respectively. The categories of the training and development sets are displayed in Table 1. Owing to the imbalance of the training set data, the official evaluation matrics is the micro-average F_1-score.

3.2 Preprocessing

For the data preprocessing, cleaning, and tokenization, as well as for most of the training sets, we used the Python Scikit-learn (Pedregosa et al., 2013) and Ekphrasis (Baziotis et al., 2017; Gimpel et al., 2011). The data is different from regular text, with substantial amounts of irregular grammar, logograms, and abbreviations, among others. Moreover, the emojis in text have an influence on the emotions detection. We studied the abbreviations of the text, determined the comparison table of abbreviations and words, and added English abbreviations to the abbreviated words. Moreover, the special emojis in text were counted, and a comparison table of expressions and corresponding explanations was generated. The processing steps are as follows:

- Process multiple consecutive punctuation points or emojis in a text into a punctuation or emoji.
- Splice the dialogue into a single text, and segment each round of dialogue with ’<eos>’.
- Use regular expressions to convert English mis-spelled words with similar rules into the correct words (for example, convert ’goooooood’ to ’good’, and ’yesssss’ to ’yes’).
- Use the Ekphrasis tool to segment texts. This tool is used to separate special emoticons (for example, convert ’:-(’ to ’unhappy face’, and ’:)’ to ’smiley face’), which is effective for the next step of expression processing.
- Traverse the word segmentation and comparison table one by one, replacing logograms, abbreviations, and emojis (for example, ’ ☹’ to ’unhappy face’ and convert ’☺’ to ’smiley face’,).

3.3 Parameter Optimization

In order to search the optimal parameters for each model, we used the Scikit-Learn toolkit to perform a grid search on the training set. In the single model tuning phase, we output the probability value of the four classifications and then performed integration. We used the micro-averaged F_1-score to evaluate the results of the soft-voting and to tune the optimal parameters.

After adjusting the single model parameters, we obtain the eight best performing models. As the training set was randomly scrambled, the training set for each cross-validation differs. Therefore, in the final integration, eight models will be run together, and the results of the eight models will be integrated to obtain the predicted results. The classifier for integrated learning is XGBoost, which uses the grid search method in Scikit-Learn to tune the optimal parameters. As the data to be learned are unbalanced, cross-validation is used to adjust the parameters. This step is applied to the training and development sets, and finally predicts the test set results.

Embeding	Dataset	Micro-Average-F_1(%)			
		Attention GRU	Capsule-Net	LSTM	GRU
GloVe	Dev	69.81	70.16	68.76	66.27
	Test	71.26	70.24	69.05	69.43
ELMo	Dev	71.36	70.18	70.32	70.12
	Test	70.04	70.95	70.92	70.1
Ensemble	Dev	76.05			
	Test	75.88			

Table 2: Final submission of development sets and tests.

	Precision	*Recall*	F_1
happy	0.717	0.687	0.701
sad	0.802	0.824	0.813
angry	0.720	0.0.819	0.766

Table 3: The result for each emotion class of test dataset.

The parameters are described as follows: model 1 and model 2 are two layers of LSTM and GRU respectively. Then there is the Dropout layer, and finally the softmax function is used to output the probability value. Model 3 and model 4 add the Attention layer and the Capsule-Net layer respectively after the stacked bidirectional GRU. The output is the same as model 1 or 2. The dropout in the GRU component is 0.25. The hidden dimension using the GloVe word vector is 120 while using the ELMO word vector is 400.The optimizer is rmsprop with a learning rate of 0.3. The loss function is categorical cross-entropy. The bacth size of model is 256. The Routings is 5 of Capsule-Net while the number of capsule is 10 and the capsule dimension is 32.

The super parameters of XGBoost are as follows: the learning rate is 0.09, the estimators are 18, the maximum depth is 4, gamma is 1.7, subsample is 0.15, colsample_bytree is 0.75, reg_alpha is 0.01, and seed is 6.

3.4 Results and Analysis

The proposed system is trained on the EmoContext training set. There are eight models, and each single model is trained using five-fold cross-validation combined with a soft-voting method. The stacking integration algorithm is used to output the final predicted results. The results of the development and test sets are presented in Table 2. the result for each emotion class of test dataset in Table 3.

It is worth noting that the results of developing the ELMo word vector on the set are significantly superior to the results of the GloVe word vector. We did not discard the GloVe word vector, and the results are not ineffective. Moreover, the performance will be improved by integrating multiple different models. Based on the results of the test set, the GloVe word vector results will be superior to those of the development set in the final prediction. In the overall comparison of the test set and development set results, the model offers a strong generalization ability.

Following the ensemble learning, our final result is 75.88%. Throughout the experiment, we found that, although we used cross-validation and applied a soft-voting mechanism, the experimental results were not very stable, the main reason for which is that the data were not balanced.

4 Conlusion

Our system won 10th place in SemEval-2019 Task 3, EmoContext: Contextual Emotion Detection in Text. The main challenge for this task was data imbalance. We achieved a competitive result using a multi-step ensemble neural network. The use of cross-validation in a single model mitigates the effects of data imbalance. A soft-voting mechanism was incorporated into the process to improve the model stability further. The results of the eight models were integrated using stacking integration. According to the final official review, our system is certainly effective. In future work, we will study how to enable the model to learn more features and improve our proposed model in the case of data imbalance.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants No.61702443 and No.61762091. We also thank anonymous reviewers for their comments.

References

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 1, pages 747–754.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Nitesh Chawla, Kevin Bowyer, Lawrence O. Hall, and W Philip Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794.

Kyunghyun Cho, Bart van Merri?nboer, Dzmitry Bahdanau, and Y Bengio. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *Computer Science*.

Chris Drummond, Robert C Holte, et al. 2003. C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II*, volume 11, pages 1–8. Citeseer.

Francois and Chollet. 2015. Keras: Deep learning library for theano and tensorflow.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments. *Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers*.

H. He and E. A. Garcia. 2008. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412—-1421.

Tomas Mikolov. 2010. Recurrent neural network based language model. *Interspeech*, 2:3.

Fabian Pedregosa, Gal Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, and Vincent Dubourg. 2013. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(10):2825–2830.

Jeffrey Pennington, Richard Socher, and Christoper Manning. 2014. GloVe: Global Vectors for Word Representation. *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532—-1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *The North American Chapter of the Association for Computational Linguistics 2019*, abs/1802.0.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. pages 3856–3866.

Nitish Srivastava Salakhutdinov, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Gary M Weiss. 2004. Mining with rarity: a unifying framework. *ACM Sigkdd Explorations Newsletter*, 6(1):7–19.

Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. 2018. Investigating Capsule Networks with Dynamic Routing for Text Classification. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3110–3119.

KDEHatEval at SemEval-2019 Task 5: A Neural Network Model for Detecting Hate Speech in Twitter

Umme Aymun Siddiqua, Abu Nowshed Chy, and Masaki Aono
Department of Computer Science and Engineering
Toyohashi University of Technology, Toyohashi, Aichi, Japan.
{aymun,nowshed}@kde.cs.tut.ac.jp and aono@tut.jp

Abstract

In the age of emerging volume of microblog platforms, especially twitter, hate speech propagation is now of great concern. However, due to the brevity of tweets and informal user generated contents, detecting and analyzing hate speech on twitter is a formidable task. In this paper, we present our approach for detecting hate speech in tweets defined in the SemEval-2019 Task 5. Our team KDEHatEval employs different neural network models including multi-kernel convolution (MKC), nested LSTMs (NLSTMs), and multi-layer perceptron (MLP) in a unified architecture. Moreover, we utilize the state-of-the-art pre-trained sentence embedding models including DeepMoji, InferSent, and BERT for effective tweet representation. We analyze the performance of our method and demonstrate the contribution of each component of our architecture.

1 Introduction

Nowadays, microblog platforms such as twitter has become the most popular communication medium among the people due to its convenient feature for sharing views, opinions, breaking news, and ideas. Besides its robust communication feature, it has facilitated the evil-minded people to propagate anti-social behavior including online harassment, cyber-bullying, and hate speech.

Hate speech is commonly defined as any communication that disparages a person or a group on the basis of some characteristics such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics (Basile et al., 2019). Given the massive amount of user-generated contents on the microblog, the problem of detecting, and therefore possibly limit the hate speech diffusion, is becoming fundamental, for instance for fighting against misogyny and xenophobia.

To address the challenges of hate speech detection in microblog platforms, Basile et al. (2019) proposed a multilingual detection of hate speech (HatEval) in twitter, task 5 at SemEval-2019. The task features two specific different targets including immigrants and women and focuses on two related subtasks.

Task A defines a two-class (or binary) classification problem where a system needs to predict whether a tweet in English or Spanish with a given target (women or immigrants) is hateful or not hateful. Whereas task B defines the aggressive behavior and target classification problem. A system first classifies a hateful tweet as aggressive or not aggressive, and then identify the target harassed as the individual or generic (i.e., single human or group). In this paper, we only focus on the English tweets for both task A and B.

The rest of the paper is structured as follows: **Section 2** provides a brief overview of prior research. In **Section 3**, we introduce our proposed neural network model. **Section 4** includes experiments and evaluations as well as the analysis of our proposed method. Some concluded remarks and future directions of our work are described in **Section 5**.

2 Related Work

Early studies on hate speech detection focused mainly on lexicon-based approaches (Kwok and Wang, 2013; Gitari et al., 2015). However, these approaches prone to failure for detecting hate speech in a microblogging platform where rare terms are evolving incessantly. Besides some researchers tackled the problem by employing feature (e.g., N-gram, TF-IDF) based supervised learning approach using SVM and Naive-Bayes classifier (Gaydhani et al., 2018; Unsvåg and Gambäck, 2018).

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 365–370
Minneapolis, Minnesota, USA, June 6–7, 2019. ©2019 Association for Computational Linguistics

More recently several researchers tried to address the problem by using state-of-the-art neural network based models. Among several prominent works, Badjatiya et al. (2017) employed multiple deep learning architectures including CNNs, LSTMs, and fastText to learn semantic word embeddings for hate speech detection. Golem et al. (2018) utilized the combination of traditional shallow machine learning models and deep learning models for hate speech detection. Pitsilis et al. (2018) utilized the ensemble of recurrent neural network (RNN) classifiers and incorporated various features associated with user-related information. Zhang et al. (2018) introduced a new method by combining a convolutional neural network (CNN) and gated recurrent unit (GRU) models. Djuric et al. (2015) used the comment embeddings for detecting hate speech.

3 Proposed Framework

In this section, we describe the details of our proposed neural network model. The goal of our proposed approach is to detect whether a tweet is hateful or not as well as determine its aggressiveness and identify the target harassed as individual or generic. We consider each task as a binary classification problem as well as train and evaluate our model accordingly. Figure 1 depicts an overview of our proposed model.

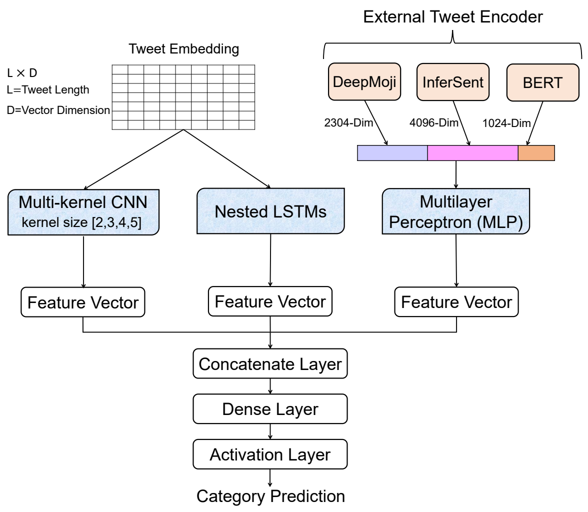


Figure 1: Proposed framework.

At first, we utilize a pre-trained word embedding model to obtain the high-quality distributed vector representations of tweets. Next, we apply the multi-kernel convolution (MKC) and nested LSTMs (NLSTMs) models to extract the higher-level feature sequences with sequential information from the tweet embeddings. Besides, we employ three different pre-trained tweet encoder models including DeepMoji, InferSent, and BERT to encode each tweet into 2304, 4096, and 1024-dimensional feature vector, respectively. These feature vectors are then combined and sent to a multi-layer perceptron (MLP) module. Finally, the generated output feature sequences from MKC, NLSTMs, and MLP are concatenated and fed into the fully-connected prediction module to determine the final category label. For the simplicity of discussion, we named our proposed neural network architecture as MKC-NLSTMs-MLP. Next, we describe each component elaborately.

3.1 Embedding Layer

Distributed representation of words known as word embedding is treated as one of the most popular representations of documents vocabulary due to its capability of capturing the context of a word within a document as well as estimating the semantic similarity and relation with other words (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2017).

In our proposed framework, we utilize a pre-trained word embedding model based on fastText (Bojanowski et al., 2017) to obtain the high-quality distributed vector representations of tweets. The dimensionality of the embedding matrix will be $L \times D$, where L is the tweet length, and D is the word-vector dimension.

3.2 Multi-kernel Convolution

The convolution layers usually applied to extract the higher level features from the given input matrix. Since kernel sizes, i.e., the size of the convolution filters have a significant effect on performance, we apply filters with different sizes to get the different kinds of effective features. In our multi-kernel convolution, We perform the convolution on the input tweet's embedding matrix by using four different kernel sizes: 2, 3, 4, and 5. Some previous studies already demonstrated the effectiveness of multi-kernel convolution over the single one (Kim, 2014; Zhang and Wallace, 2015; Wang et al., 2017).

3.3 Nested LSTMs

In nested LSTMs (NLSTMs) (Moniz and Krueger, 2017), the LSTM memory cells have access to their inner memory, where they can selectively read and write relevant long-term information. While the value of the outer memory cell in the LSTM is estimated as $c_t^{outer} = f_t \odot c_{t-1} + i_t \odot g_t$, memory cells of the NLSTM use the concatenation $(f_t \odot c_{t-1}, i_t \odot g_t)$ as input to an inner LSTM (or NLSTM) memory cell, and set $c_t^{outer} = h_t^{inner}$. The inner memories of NLSTMs operate on longer time-scales and capture the context information from the input tweets effectively.

3.4 Pre-trained Models for Feature Encoding

In order to extract features for effective tweet representation, we utilize the three state-of-the-art pre-trained sentence embedding model including DeepMoji, BERT, and InferSent. In this section, we briefly describe these models.

DeepMoji: DeepMoji (Felbo et al., 2017) performs distant supervision on a dataset of 1246 million tweets comprising a more diverse set of noisy labels. DeepMoji uses an embedding layer of 256 dimensions to project each word of a tweet into a vector space. Two bidirectional LSTM layers with 1024 hidden units in each (512 in each direction) are applied to capture the context of each word. Finally, an attention layer takes all of these layers as input using skip-connections. We employ the representation vector of dimension 2304 obtained from the attention layer as the features.

BERT: BERT (Devlin et al., 2018) stands for Bidirectional Encoder Representations from Transformers, which is a new method of pre-training sentence representations. We employ the BERT-Large, Uncased model to encode each tweet into a 1024-dimensional feature vector.

InferSent: InferSent (Conneau et al., 2017) is a universal sentence embedding model trained using the supervised data of the Stanford Natural Language Inference (SNLI) datasets. We employ the InferSent model trained on fastText (Bojanowski et al., 2017) vectors to encode each tweet into a 4096-dimensional feature vector.

3.5 Multi-layer Perceptron

After extracting features from the pre-trained external tweet encoder model, we concatenate them and pass to a fully connected multi-layer perceptron (MLP) network.

A multilayer perceptron (MLP) (Pedregosa et al., 2011) is a feed-forward artificial neural network model that maps sets of input data onto a set of appropriate outputs. An MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron with a nonlinear activation function. MLP utilizes a supervised learning technique called back-propagation for training the network.

3.6 Prediction Module and Model Training

We concatenate the final tweet representation from the multi-kernel convolution (MKC) module, NLSTMs module, and MLP module and pass it to a fully connected softmax layer for category prediction. We consider cross-entropy as the loss function and train the model by minimizing the error, which is defined as:

$$E(x^{(i)}, y^{(i)}) = \sum_{j=1}^{k} 1\{y^{(i)} = j\} \log(y_j^{\sim(i)})$$

where $x^{(i)}$ is the training sample with its true label $y^{(i)}$. $y_j^{\sim(i)}$ is the estimated probability in $[0, 1]$ for each label j. $1\{condition\}$ is an indicator which is 1 if true and 0 otherwise. We use the stochastic gradient descent (SGD) to learn the model parameter and adopt the Adam optimizer (Kingma and Ba, 2014).

4 Experiments and Evaluations

4.1 Dataset Collection

The multilingual detection of hate speech (HatEval) task 5 at SemEval-2019 (Basile et al., 2019) provides a benchmark dataset to evaluate the performance of the participants' systems. The proposed task features two specific different targets including immigrants and women in a multilingual perspective, for Spanish and English. However, we only used the English dataset to evaluate our proposed system. The training, validation, and test set of the English dataset contains the 9000, 1000, and 2971 annotated tweets, respectively.

4.2 Model Configuration

In the following, we describe the set of parameters that we have used to design our proposed neural network model, MKC-NLSTMs-MLP. Our designed model was based on Tensorflow (Abadi et al., 2016) and trained on a GPU (Owens et al., 2008) to capture the benefit from the efficiency of parallel computation of tensors. We performed hyper-parameter optimization using a simple grid search. We used the 300-dimensional fastText embedding model pre-trained on Wikipedia with skip-gram (Bojanowski et al., 2017) to initialize the word embeddings in the embedding layer described in Section 3.1. For the multi-kernel convolution described in Section 3.2, we employed 4 different kernel sizes including (2,3,4,5), and the number of filters was set to 600. The nested LSTMs module contains 2 layers and multi-layer perceptron (MLP) module contains 3 fully-connected dense layers. We trained all models with 15 epochs with a batch size of 32 and an initial learning rate 0.001 by Adam optimizer. The MLP layers were dropped out with a probability of 0.02. $L2$ regularization with a factor of 0.01 was applied to the weights in the softmax layer. Unless otherwise stated, default settings were used for the other parameters.

4.3 Evaluation Measures

To evaluate the performance of the system, the organizers used different strategies and metrics for the task A and B (Basile et al., 2019). For the task A, standard evaluation metrics, including accuracy, precision, recall, and F1-score were applied to estimate the performance of a system. However, F1-score considered as the primary evaluation measure for this task.

For the task B, macro average $F1$-score of the hate speech (HS), target range (TR), and aggressiveness (AG) category and exact match ratio (EMR) of these categories are used as the evaluation measures. EMR considered as the primary evaluation measure for task B.

4.4 Experimental Results

We now evaluate the performance of our proposed method, MKC-NLSTMs-MLP, in this section. The summarized results for task A and task B are presented in Table 1 and Table 2, respectively.

At first, we presented the performance of our proposed method denoted by team name KDEHatEval as well as presenting the performance of randomly chosen top-ranked participated systems and

Team Name	Accuracy	Precision	Recall	F1-Score
KDEHatEval	0.493	0.633	0.555	0.440
Fermi	0.653	0.690	0.679	0.651
YNU_DYX	0.560	0.636	0.603	0.546
SINAI_DL	0.535	0.601	0.577	0.519
Hateminers	0.544	0.658	0.596	0.516
SVC Baseline	0.492	0.595	0.549	0.451
MFC Baseline	0.579	0.289	0.500	0.367

Table 1: (Task A) Our result with other selected teams.

Team Name	Avg. F1-Score	Exact Match Ratio (EMR)
KDEHatEval	0.559	0.324
MFC Baseline	0.421	0.580
CIC-1	0.551	0.568
SINAI_DL	0.611	0.384
Hateminers	0.589	0.357
SVC Baseline	0.578	0.308

Table 2: (Task B) Our result with other selected teams.

HatEval-2019 baselines. The organizers used the SVC (a linear support vector machine) and MFC (a trivial model that assigns the most frequent label, estimated on the training set) as the baseline system (Basile et al., 2019).

In order to estimate the effect of each component of our MKC-NLSTMs-MLP model, we performed the component ablation study. In this regard, we removed one component each time and repeated the experiment. The summarized experimental results of component ablation study for the task A are presented in Table 3.

Method	Accuracy	Precision	Recall	F1-Score
MKC-NLSTMs-MLP	0.493	0.633	0.555	0.440
−MKC	0.441	0.507	0.503	0.381
−NLSTMs	0.483	0.630	0.548	0.423
−MLP	0.495	0.636	0.557	0.443
−MKC−NLSTMs	0.458	0.507	0.505	0.431
−MKC−MLP	0.475	0.592	0.537	0.419
−NLSTMs−MLP	0.485	0.640	0.549	0.423

Table 3: (Task A) Ablation study of our proposed method.

From the results, it can be observed that when removing multi-kernel convolution (MKC) and nested LSTMs (NLSTMs) the overall F1 score decreased by 5.6% and 1.7%, respectively. However, when removing external tweet embedding with MLP module, the results increased by 0.3%. This observation deduced that in our current architecture, external pre-trained model features with MLP contributed negatively.

Besides, removing one component, we also perform ablation study by removing two components and present the results accordingly in Table 3. This analysis provides the overall performance of the individual component.

5 Conclusion

In this paper, we presented our approach to the SemEval-2019 Task 5: HatEval: Detection of hate speech. We tackled the problem by employing several deep learning techniques including multi-kernel convolution, nested LSTMs, and multi-layer perceptron in a unified architecture.

Though we have used the state-of-the-art techniques in our proposed approach, the overall performance is not satisfactory. The contribution of nested LSTMs (NLSTMs) is not significant while compared with the multi-kernel convolution (MKC). Regarding this, one possible solution will be used the MKC on top of NLSTMs. Moreover, we observed that the multi-layer perceptron (MLP) model trained with the concatenated features from the pre-trained sentence embedding models hurts the performance of our proposed architecture. We need to observe the ablation study of the sentence embedding models as well as modify the MLP architecture to mitigate this issue.

Therefore, there is much room left to improve the performance of our method presented in HatEval-2019 task. In the future, we have a plan to overcome these limitations by introducing several sophisticated techniques.

Acknowledgments

The part of this research is supported by MEXT KAKENHI, Grant-in-Aid for Scientific Research (B), Grant Number 17H01746 and by grants from the KDDI Foundation.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI)*, pages 265–283. USENIX Association.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web (WWW) Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics (TACL)*, 5:135–146.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680, Copenhagen, Denmark. ACL.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*, pages 29–30. ACM.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1615–1625. ACL.

Aditya Gaydhani, Vikrant Doma, Shrikant Kendre, and Laxmi Bhagwat. 2018. Detecting hate speech and offensive language on twitter using machine learning: An n-gram and tfidf based approach. *arXiv preprint arXiv:1809.08651*.

Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long. 2015. A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering (IJMUE)*, 10(4):215–230.

Viktor Golem, Mladen Karan, and Jan Šnajder. 2018. Combining shallow and deep learning for aggressive text detection. In *Proceedings of the 1st Workshop on Trolling, Aggression, and Cyberbullying (TRAC-2018)*, pages 188–198.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. ACL.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Irene Kwok and Yuzhou Wang. 2013. Locate the hate: detecting tweets against blacks. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, pages 1621–1622. AAAI Press.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3111–3119.

Joel Ruben Antony Moniz and David Krueger. 2017. Nested lstms. In *Proceedings of the 9th Asian Conference on Machine Learning (ACML)*, pages 530–544. Springer.

John D Owens, Mike Houston, David Luebke, Simon Green, John E Stone, and James C Phillips. 2008. Gpu computing. *Proceedings of the IEEE*, 96(5):879–899.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research (JMLR)*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. ACL.

Georgios K Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*.

Elise Fehn Unsvåg and Björn Gambäck. 2018. The effects of user features on twitter hate speech detection. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 75–85.

Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. 2017. Combining knowledge with deep convolutional neural networks for short text classification. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2915–2921. AAAI Press.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting hate speech on twitter using a convolution-gru based deep neural network. In *Proceedings of the 15th European Semantic Web Conference (ESWC)*, pages 745–760. Springer.

ABARUAH at SemEval-2019 Task 5 : Bi-directional LSTM for Hate Speech Detection

Arup Baruah
Dept. of Comp. Sc. & Engg.
IIIT Guwahati, India
arup.baruah@gmail.com

Ferdous Ahmed Barbhuiya
Dept. of Comp. Sc. & Engg.
IIIT Guwahati, India
ferdous@iiitg.ac.in

Kuntal Dey
IBM Research India
New Delhi
kuntadey@in.ibm.com

Abstract

In this paper, we present the results obtained using bi-directional long short-term memory (BiLSTM) with and without attention and Logistic Regression (LR) models for SemEval-2019 Task 5 titled ”HatEval: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter”. This paper presents the results obtained for Subtask A for English language. The results of the BiLSTM and LR models are compared for two different types of preprocessing. One with no stemming performed and no stopwords removed. The other with stemming performed and stopwords removed. The BiLSTM model without attention performed the best for the first test, while the LR model with character n-grams performed the best for the second test. The BiLSTM model obtained an F1 score of 0.51 on the test set and obtained an official ranking of 8/71.

1 Introduction

Davidson et al. (2017) has defined hate speech as “language that is used to express hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group.” Gambäck and Sikdar (2017), Badjatiya et al. (2017), Waseem (2016) and Waseem et al. (2017) have used the term hate speech to indicate tweets having racist or sexist comments. Social media is becoming a convenient medium to spread hate speech. Hate speech spread through social media has fueled riots in Myanmar [1], Sri Lanka [2], Charlottesville (USA) [3], and many other parts of the world. Thus, it is becoming increasingly important to detect and remove hate messages from the web. It is not possible to manually moderate the vast amount of text exchanged on the web. Developing automated systems to recognize hate speech is becoming crucially important. However, detecting hate speech in a text is more than just checking for the presence of hate words. Lexicon based approaches have not been very effective in hate speech detection (Nobata et al., 2016).

As part of the 13th workshop on semantic evaluation (SemEval-2019), shared task 5 defines two subtasks with regard to detection of hate speech against immigrants and women in Twitter (Basile et al., 2019). This task was conducted for tweets in English and Spanish language. In Subtask A, it is required to determine if a tweet, with a given target, is hateful or not. In Subtask B, it is required to determine if a given hateful tweet is aggressive or not and whether it targets an individual or a group.

2 Related Work

Nobata et al. (2016) studied the performance of different features such as character n-grams, word n-grams, word2vec, character2vec, etc. in detecting hate speech. A regression model was used in their study. Malmasi and Zampieri (2017) made a similar study to compare the performance of different features in detecting hate speech. Djuric et al. (2015) used paragraph embeddings for detecting hate speech. Wulczyn et al. (2017) worked on detecting insults in Wikipedia comments. Davidson et al. (2017) worked on detecting hate speech when hate words are not explicitly used in the text. Malmasi and Zampieri (2018) used ensemble method and combined 16 different base classifiers to detect hate speech. Serrà et al. (2017) used character-based Recurrent Neural Network (RNN) to study the use of out-of-vocabulary words in hate speech. Gao and Huang (2017) used BiLSTMs with attention mechanism

[1]https://www.nytimes.com/2018/10/15/technology/myanmar-facebook-genocide.html

[2]https://qz.com/1223787/sri-lanka-shut-down-facebook-whatsapp-and-instagram-to-stop-anti-muslim-violence/

[3]https://psmag.com/social-justice/how-social-media-helped-organize-and-radicalize-americas-newest-white-supremacists

to detect hate speech. Pavlopoulos et al. (2017) used Convolutional Neural Network (CNN) and RNN with attention mechanism to moderate user comments. Sax (2016) compared the performance of several deep learning techniques, LR and Support Vector Machine (SVM) models in detecting hate speech.

3 Data

Table 1 below shows the proportion of positive and negative instances of hate speech in the train, development and test data sets. As can be seen, 42% of the instances in each of the data set are hate speech. The data collected in Gao et al. (2017) had only 0.6% hateful tweets. Nobata et al. (2016) found that only 5.9% of the online comments contained hate speech. The data sets used for this task, however, are quite balanced.

Type	Not Hate Speech	Hate Speech	Total
Train	5217 (58%)	3783 (42%)	9000
Dev	573 (57.3%)	427 (42.7%)	1000
Test	1718 (57.85%)	1252 (42.15%)	2970

Table 1: Data set statistics

The models used in our study were trained and validated using the train and development sets provided as part of this task. No other external data sets were used for training or validating. However, pre-trained GloVe [4] word vectors trained using 2 billion tweets were used as features for the two BiLSTM models. The 200-dimensional word vectors were used in our experiments.

4 Experimental Settings

4.1 Preprocessing

The preprocessing performed on the text includes the following -

1. All URLs, mentions and non-alphabetic characters were removed from the tweets.
2. The tweets were then converted to lowercase.
3. Stemming was performed using NLTK's Lancaster stemmer.
4. Stopwords were removed.
5. Tokenizer was used to convert each tweet into a sequence of integers by replacing each token by its index into the vocabulary. The tweet with the maximum length had 58 tokens (when stopwords were retained). This value is later used as the length of the input sequences to the Embedding layer of the BiLSTM models. So, tweets with length less than 58 were padded with zeros, so as to make all the tweets to be of the same length.

[4]https://nlp.stanford.edu/projects/glove/

4.2 Models Used

In our study, we used a BiLSTM without attention, and a BiLSTM with attention and an LR model. The details of our models are provided below.

4.2.1 BiLSTM without attention mechanism

Figure 1 shows the architecture of the BiLSTM used in our study. Both the forward and backward Long Short-Term Memory (LSTM) layers of the bi-directional layer of this model consisted of 100 units. Pre-trained GloVe embeddings were used to train the model. An embedding layer was used to feed the word vectors to the BiLSTM layer. A dropout of 0.25 was applied to the input of LSTM and a dropout of 0.1 was used for the recurrent connections. The BiLSTM layer was set to return the hidden state for each timestep. Thus, the output shape of the BiLSTM layer was (None,58,200). A global max pooling layer was used on top of the BiLSTM layer. This resulted in an output of the shape (None,200). The output from the global max pooling layer was fed to a Dense layer having 100 units. The Rectified Linear Unit (ReLU) activation function was used for this Dense layer. The output of the Dense layer was of the shape (None,100). The output was passed through a dropout layer with the rate set to 0.25. The output from the dropout layer was then passed through another Dense layer having a single unit. The sigmoid activation function was used for this layer. The model was trained using the Adam optimizer. The loss function used was binary cross-entropy. The model was trained with a batch size set to 32. The hyperparameter values used for the model are summarized in Table 2.

4.2.2 BiLSTM with attention mechanism

This model is exactly the same as the model described in 4.2.1 except that the global max pooling layer was replaced with attention mechanism. The hyperparameter values for both the models were also same.

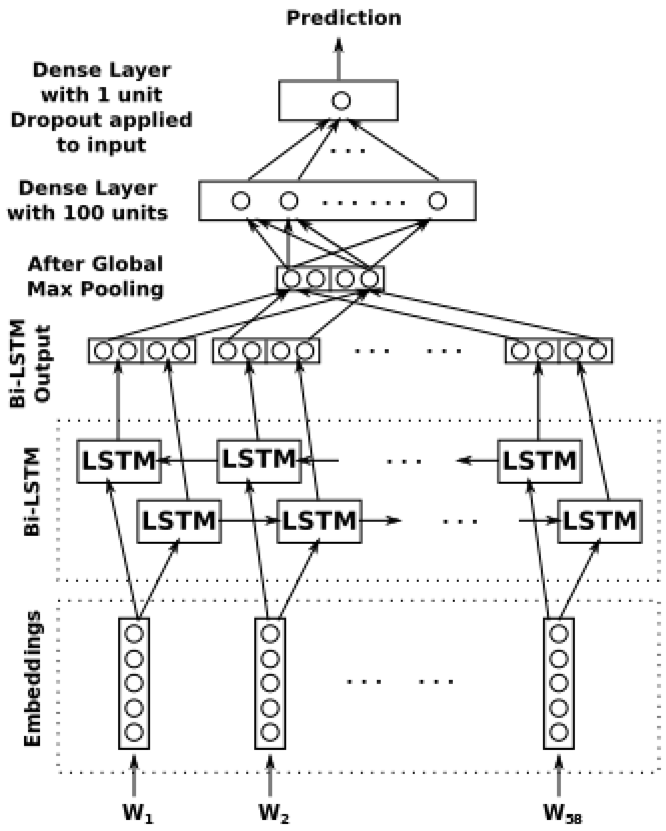


Figure 1: Architecture of the BiLSTM model used

Parameter	Value
Number of LSTM units	100
LSTM dropout	0.25
Recurrent dropout	0.10
Units in 1st Dense layer	100
Activation Function for 1st Dense layer	ReLU
Rate for dropout layer	0.25
Units in 2nd Dense layer	1
Activation Function for 2nd Dense layer	sigmoid
Optimizer	Adam
Loss Function	Binary cross-entropy

Table 2: Hyperparameters for the BiLSTM model

4.2.3 Logistic Regression

The third model we used was an LR model with L2 regularization. The class weight and C parameter were set to 'balanced' and 1.2 respectively. This model was trained with character n-grams (1 to 6), word n-grams (1 to 3), and a combination of both character and word n-grams. The hyperparameter values for the LR model were set as shown in Table 3.

5 Results & Discussions

The following tests were performed by us:

1. BiLSTM model trained using GloVe word embeddings as features.

2. BiLSTM model with attention mechanism trained using GloVe word embeddings as features.

3. LR model trained using character n-grams (1 to 6) only.

4. LR model trained using word n-grams (1 to 3) only.

5. LR model trained using both character and word n-grams concatenated together.

Parameter	Value
Regularization	L2
C	1.2
Class weight	balanced

Table 3: Hyperparameters for the LR model

Table 4 shows the results obtained by our models on the development set when stemming was not performed and stopwords were not removed. Table 5 shows the results when stemming was performed and stopwords were also removed.

Approach	Features	Acc	Prec	Rec	F1
BiLSTM	GloVe word embeddings	**0.748**	**0.749**	**0.748**	**0.748**
BiLSTM with attention	GloVe word embeddings	0.737	0.738	0.737	0.737
Logistic Regression	Char n-grams (1 to 6)	0.727	0.733	0.727	0.728
Logistic Regression	Word n-grams (1 to 3)	0.722	0.729	0.722	0.723
Logistic Regression	Char n-grams & Word n-grams	0.727	0.734	0.727	0.728

Table 4: Results of our models on Dev set of Task 5-Subtask A (No stemming + No Stopwords removed)

Approach	Features	Acc	Prec	Rec	F1
BiLSTM	GloVe word embeddings	0.674	0.699	0.674	0.675
BiLSTM with attention	GloVe word embeddings	0.698	0.698	0.698	0.689
Logistic Regression	Char n-grams (1 to 6)	**0.743**	**0.749**	**0.743**	**0.744**
Logistic Regression	Word n-grams (1 to 3)	0.727	0.731	0.727	0.728
Logistic Regression	Char n-grams & Word n-grams	0.739	0.746	0.739	0.740

Table 5: Results of our models on Dev set of Task 5-Subtask A (Stemming + Stopwords removed)

Approach	Accuracy	Precision	Recall	F1
SVC Baseline	0.49	0.60	0.55	0.45
MFC Baseline	**0.58**	0.29	0.50	0.37
BiLSTM	0.54	**0.64**	**0.59**	**0.51**
Best System	0.65	0.69	0.68	0.65

Table 6: Official results for Task 5-Subtask A

As can be seen from Table 4, the BiLSTM model without attention outperformed the other two models for all the metrics. However, the improvements achieved were not very significant compared to the other two models. It can also be seen that the choice of character or word n-grams did not make much difference to the performance of the LR model. Equivalent results were obtained for all the 3 tests performed using the LR model. This is surprising considering the fact that character n-grams usually performs better than word n-grams for text containing obfuscated words. Mehdad and Tetreault (2016) mentions that offenders often obfuscate the hate words in order to avoid detection by keyword-based filters. So, character n-gram features should have improved the performance of the model. One explanation for this observation could be the removal of numeric and special characters from the tweets during the data preprocessing stage. Numeric and special characters are used frequently used to obfuscate hate words. 'ass' replaced by 'a$$', 'slut' replaced by 's1ut' etc. are examples of such obfuscation. So, a test was performed without removing the numeric and special characters. However, no significant increase in the performance of character-based model was observed.

As can be seen from Table 5, the LR models performed better than the BiLSTM models when stemming was performed and stopwords were removed. The character n-gram based LR model performed the best for all the metrics considered.

The predictions obtained for the test set using the BiLSTM model without attention mechanism were submitted as the final predictions. The model that was trained with stopwords retained and stemming not performed was used to make the predictions on the test set. The official results obtained for the submission are shown in Table 6. Our official ranking is 8/71 in subtask A for the English language. As can be seen from the results, the MFC baseline had the best accuracy score. By labeling all the test instances with the most frequent label, the MFC baseline was able to obtain a better accuracy score. The MFC baseline achieved a better accuracy score at the cost of a low precision value. This resulted in a low F1 score for the baseline. Our BiLSTM model obtained a significantly higher F1 score compared to the MFC baseline. Our BiLSTM model outperformed the SVC baseline on all the metrics.

Table 7 shows the confusion matrices for the tests performed by retaining the stopwords and without performing stemming. The BiLSTM models were able to achieve better F1 score compared to the LR model by making better predictions for the benign class.

Table 8 shows the confusion matrices for the tests performed with stopwords removed and stemming performed. As can be seen, the BiLSTM models with attention and without attention show opposite tendencies. While the BiLSTM model without attention gets better in predicting the hate class, it becomes weaker in predicting the benign class. The opposite is true for the BiLSTM model with attention. The character n-gram based LR model gets better in predicting both the hate and benign classes.

6 Error Analysis

From the errors made by our system, it is evident that the model was not able to determine correctly if the hate words have really been used to express hate. For e.g., the following tweet from the test set is not a hate speech - "I can be a bitch and

	BiLSTM Predictions		**BiLSTM with Attention Predictions**		**LR (char) Predictions**		**LR (word) Predictions**		**LR (char+word) Predictions**	
	Not	**Hate**	**Not**	**Hate**	**Not**	**Hate**	**Not**	**Hate**	**Not**	**Hate**
Not	437	136	438	135	411	162	408	165	410	163
Hate	116	311	128	299	111	316	113	314	110	317

Table 7: Confusion Matrix (No Stemming + No stopword removed)

	BiLSTM Predictions		**BiLSTM with Attention Predictions**		**LR (char) Predictions**		**LR (word) Predictions**		**LR (char+word) Predictions**	
	Not	**Hate**	**Not**	**Hate**	**Not**	**Hate**	**Not**	**Hate**	**Not**	**Hate**
Not	345	228	476	97	421	152	418	155	416	157
Hate	98	329	205	222	105	322	118	309	104	323

Table 8: Confusion Matrix (Stemming + Stopwords removed)

an asshole but I will love you and care about you more than any other person you have met." Here, the speaker is attributing the word 'bitch' to himself/herself. So, the tweet cannot be a hate speech. However, our system wrongly classifies the tweet as a hate speech.

There have been many instances where our system wrongly classifies a tweet as hate speech just because of the mere presence of words such as 'bitch', '#buildthewall' etc. even when the tweet is not intended against women or immigrants. For e.g., the tweet 'He is a snake ass bitch. He is a fugly slut...' is a hate speech but it is not intended against women. But our system was not able to detect this and wrongly classifies it.

7 Conclusion

With hate speech in social media fomenting many riots in different parts of the world, it is becoming increasingly important to prevent their spread. While manual moderation is almost impossible, the need of the time is automated systems for their removal. The BiLSTM and logistic regression models used in this study have obtained some success compared to the baselines used. But there is much left to be desired. Hate words used in the benign sense and hate speech not directed at women and immigrants were wrongly getting classified. Contextual information and features such as part-of-speech (POS), dependency relations may help in classifying such instances correctly.

References

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep Learning for Hate Speech Detection in Tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760, Perth.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of the Eleventh International AAAI Conference on Web and Social Media (ICWSM 2017)*, pages 512–515, Montreal.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate Speech Detection with Comment Embeddings. In *International World Wide Web Conference (WWW), 2015*, pages 29–30, Florence, Italy.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using convolutional neural networks to classify hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90, Vancouver, BC, Canada. Association for Computational Linguistics.

Lei Gao and Ruihong Huang. 2017. Detecting online hate speech using context aware models. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 260–266, Varna, Bulgaria. INCOMA Ltd.

Lei Gao, Alexis Kuppersmith, and Ruihong Huang. 2017. Recognizing explicit and implicit hate speech using a weakly supervised two-path bootstrapping approach. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 774–782, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting hate speech in social media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 467–472, Varna, Bulgaria. INCOMA Ltd.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in discriminating profanity from hate speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30(2):187–202.

Yashar Mehdad and Joel Tetreault. 2016. Do characters abuse more than words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303, Los Angeles. Association for Computational Linguistics.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive Language Detection in Online User Content. In *Proceedings of the 25th International Conference on World Wide Web*, pages 145–153, Montreal.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. Deep learning for user comment moderation. In *Proceedings of the First Workshop on Abusive Language Online*, pages 25–35, Vancouver, BC, Canada. Association for Computational Linguistics.

S. Sax. 2016. Flame Wars: Automatic Insult Detection. `http://cs224d.stanford.edu/reports/Sax.pdf`, (Technical Report).

Joan Serrà, Ilias Leontiadis, Dimitris Spathis, Gianluca Stringhini, Jeremy Blackburn, and Athena Vakali. 2017. Class-based prediction errors to detect hate speech with out-of-vocabulary words. In *Proceedings of the First Workshop on Abusive Language Online*, pages 36–40, Vancouver, BC, Canada. Association for Computational Linguistics.

Zeerak Waseem. 2016. Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 138–142, Austin, Texas. Association for Computational Linguistics.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding abuse: A typology of abusive language detection subtasks. In *Proceedings of the First Workshop on Abusive Language Online*, pages 78–84, Vancouver, BC, Canada. Association for Computational Linguistics.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex Machina: Personal Attacks Seen at Scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399, Perth.

Amobee at SemEval-2019 Tasks 5 and 6: Multiple Choice CNN Over Contextual Embedding

Alon Rozental[*], **Dadi Biton**[*]
Amobee Inc., Tel Aviv, Israel
{alon.rozental,dadi.biton}@amobee.com

Abstract

This article describes Amobee's participation in "HatEval: Multilingual detection of hate speech against immigrants and women in Twitter" (task 5) and "OffensEval: Identifying and Categorizing Offensive Language in Social Media" (task 6). The goal of task 5 was to detect hate speech targeted to women and immigrants. The goal of task 6 was to identify and categorized offensive language in social media, and identify offense target. We present a novel type of convolutional neural network called "Multiple Choice CNN" (MC-CNN) that we used over our newly developed contextual embedding, Rozental et al. (2019)[1]. For both tasks we used this architecture and achieved 4th place out of 69 participants with an F_1 score of 0.53 in task 5, in task 6 achieved 2nd place (out of 75) in Sub-task B - automatic categorization of offense types (our model reached places 18/2/7 out of 103/75/65 for sub-tasks A, B and C respectively in task 6).

1 Introduction

Offensive language and hate speech identification are sub-fields of natural language processing that explores the automatic inference of offensive language and hate speech with its target from textual data. The motivation to explore these sub-fields is to possibly limit the hate speech and offensive language on user-generated content, particularly, on social media. One popular social media platform for researchers to study is Twitter, a social network website where people "tweet" short posts. Each post may contain URLs and/or mentions of other entities on twitter. Among these "tweets" we can find various opinions of people regarding political events, public figures, products, etc. Hence, Twitter data turned into one of the main data sources for both academia and industry. Its unique insights are relevant for business intelligence, marketing and e-governance. This data also benefits NLP tasks such as sentiment analysis, offensive language detection, topic extraction, etc.

Both the OffensEval 2019 task (Zampieri et al. (2019b)) and HatEval 2019 task are part of the SemEval-2019 workshop. OffensEval has 3 sub-tasks with over 65 groups who participate in each sub-task and HatEval has 2 sub-tasks with 69 groups.

Word embedding is one of the most popular representations of document vocabulary in low-dimensional vector. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc. For this work, word embedding was created with a model similar to Bidirectional Encoder Representations from Transformers (BERT), Devlin et al. (2018). BERT is a language representation model designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT representations can be fine-tuned to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications. Besides the word embedding, BERT generates a classification token, which can be used for text classification tasks.

This paper describes our system for the OffensEval 2019 and HatEval 2019 tasks, where our new contribution is the use of contextual embedding (modified BERT) together with an appropriate network architecture for such embeddings .

The paper is organized as follows: Section 2 describes the datasets we used and the pre-process

[*]These authors contributed equally to this work.
[1]To be published.

Sub-Task A		Sub-Task B		Sub-Task C	
Label	Train	Label	Train	Label	Train
Offensive	4,400	Targeted	3,876	Group	1,074
Not offensive	8,840	Not targeted	524	Individual	2,407
				Other	395
Total	13,240	Total	4,400	Total	3,876
(a)		(b)		(c)	

Table 1: Distributions of lables in OffensEval 2019.

phase. Section 3 describes our system architecture and presents the MC-CNN. In section 4 we present the results of both tasks - the OffensEval and HatEval. Finally, in section 5 we review and conclude the system.

2 Data and Pre-Processing

We used Twitter Firehose dataset. We took a random sample of 50 million unique tweets using the Twitter Firehose service. The tweets were used to train language models and word embeddings; in the following, we will refer to this as the Tweets_50M dataset.

A modified language model, based on BERT, was trained using a large Tweets_50M dataset, containing 50 million unique tweets. We trained two models, one used to predict hate speech in posts (task 5) and the other used to predict offensive language in posts (task 6). The pre-process on the Tweets_50M dataset consists of replacing URLs and Twitter handles with special tokens and keeping only the first 80 sub-word tokens in each tweet (for our vocabulary over 99% of the tweets contain less than 80 tokens).

The language model we trained differs from Devlin et al. (2018) mainly by the addition of a latent variable that represents the topic of the tweet and the persona of the writer. The work on this model is still in progress and in this work we have used an early version of the model described in Rozental et al. (2019).

2.1 OffensEval

OffensEval 2019 is divided into three sub-tasks.

1. Sub-task A - Offensive language identification - identify whether a text contains any form of non-acceptable language (profanity) or a targeted offense.

2. Sub-task B - Automatic categorization of offense types - identify whether a text contains targeted or non-targeted profanity and swearing.

3. Sub-task C - Offense target identification - determine whether the target of the offensive text is an individual, group or other (e.g., an organization, a situation, an event, or an issue).

The official OffensEval task datasets, retrieved from social media (Twitter). Table 1 presents the label distribution for each sub-task. For further shared task description, data-sets and results regarding this task, see Zampieri et al. (2019a).

2.2 HatEval

HatEval 2019 is divided into two sub-tasks.

1. Sub-task A - Hate Speech Detection against Immigrants and Women: a two-class classification where systems have to predict whether a tweet in English with a given target (women or immigrants) is hateful or not hateful.

2. Sub-task B - Aggressive behavior and Target Classification: where systems are asked first to classify hateful tweets for English and Spanish (e.g., tweets where Hate Speech against women or immigrants has been identified) as aggressive or not aggressive, and second to identify the target harassed as individual or generic (i.e. single human or group). In this paper we will focus only on sub-task A as none of the participants overcame the baseline accuracy in sub-task B.

There were 69 groups who participated in sub-task A. Table 2 presents the label distribution in sub-

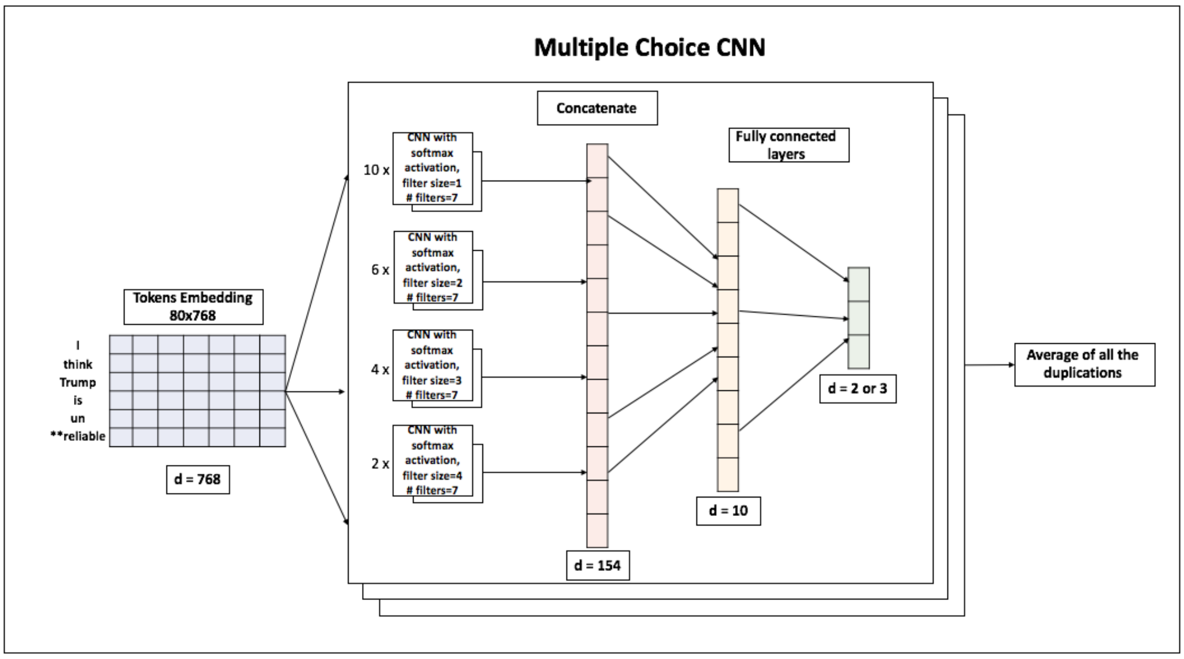

Multiple Choice CNN
Concatenate
Fully connected layers
10 x
CNN with softmax activation, filter size=1 # filters=7
6 x
CNN with softmax activation, filter size=2 # filters=7
4 x
CNN with softmax activation, filter size=3 # filters=7
2 x
CNN with softmax activation, filter size=4 # filters=7
Tokens Embedding 80x768
I
think
Trump
is
un
**reliable
d = 768
d = 154
d = 10
d = 2 or 3
Average of all the duplications

Figure 1: Architecture Of Amobee Offensive Language Detector.

Label	Train
Hate speech	4,210
Not hate speech	5,790
Total	10,000

Table 2: Distributions Of Labels In HatEval 2019.

task A. For further shared task description, datasets and results regarding this task, see Basile et al. (2019). HatEval also included Spanish task which we didn't participate in.

3 Multiple Choice CNN

For both tasks, using our contextual word embedding, we tried several basic approaches - A feed forward network using the classification vector and an LSTM and simple CNNs Zhang and Wallace (2015) using the words vectors. These approaches overfitted very fast, even for straightforward unigram CNN with 1 filter, and their results were inferior to those obtained by similar models over a Twitter specific, Word2Vec based embedding, Mikolov et al. (2013); Rozental et al. (2018). The fast overfitting is due to the information contained in contextual embedding which was not reflected in Word2Vec based embedding.

In order to avoid overfitting and achieve better results we created the MC-CNN model. The motivation behind this model is to replace quantitative questions such as "how mad is the speaker?", where the result is believed to be represented by the activation of the corresponding filter, with multiple choice questions such as "what is the speaker - happy/sad/other?", where the number of choices denoted by the number of filters. By forcing the sum of the filter activations for each group to be equal to 1, we believe that we have acheived this effect.

The model that produced the best results is an ensemble of multiple MC-CNNs over our developed contextual embedding, described in figure 1. On top of our contextual embedding, we used four filter sizes - 1-4 sub-word token n-grams. For each filter size individual filters were divided into groups of 7 and a softmax activation applied on the output of each group. These outputs were concatenated and passed to a fully connected feed forward layer of size 10 with tanh activation before it yeiled the networks' prediction. To decrease the variance of the results, there were multiple duplications of this architecture, where the final prediction was the average of all the duplications' output.

4 Results

We chose to use this architecture for both tasks because we believe that the BERT model output contains most of the information about the tweet. The layers above, the MC-CNN and the fully connected layers, adapt it to the given task. We think that this model can be use for variety of NLP tasks in twitter data with the appropriate hyper-parameters tuning.

The results yielded from the architecture which was described in figure 1 for both tasks. We optimized the hyper-parameters to maximize the F_1score using categorical cross entropy loss. The tuned parameters were the activation function of the filters and the number of filters in the MC-CNNs, the size of the filter groups of the MC-CNN, and the hidden layer size. The best result were achieved with a sigmoid activation function on the filters, where the number of filters was 7 in each group. There were 10, 6, 4 and 2 filter groups for filter sizes of 1, 2, 3 and 4 respectively. The model with those hyper-parameters yielded the best results in both tasks.

At HatEval the model achieved an F_1score of 0.535. In table 3 there is the best result compared to two baselines- linear Support Vector Machine based on a TF-IDF representation (SVC), and a trivial model that assigns the most frequent label (MFC), according to the F_1score.

System	F1 (macro)	Accuracy
SVC baseline	0.451	0.492
MFC baseline	0.367	0.579
MC-CNN	**0.535**	**0.558**

Table 3: F_1Score And Accuracy Of MC-CNN Comparing To Baselines At HatEval.

At OffensEval the model achieved an F_1score of 0.787, 0.739 and 0.591 for sub-tasks A, B and C respectively. In table 4 there is the best result compared to the baseline for sub-tasks A, B and C respectively according to the F_1score.

5 Conclusion

In this paper we described the system Amobee developed for the HatEval and OffensEval tasks. It consists of our novel task specific contextual embedding and MC-CNNs with softmax activation. The use of social networks motivated us to train contextual embedding based on the Twitter dataset, and use the information learned in this language model to identify offensive language and hate speech in the text. Using MC-CNN helped overcome the overfitting caused by the embedding. In order to decrease the variance of the system we used duplications of this model and averaged the results. This system reached 4th place at the HateEval task with an F_1score of 0.535, and 2nd place at sub-task B in the OffensEval task, with an F_1score of 0.739. As we mentioned, we used an early version of a Twitter specific language model to achieve the above results. We plan to release the complete, fully trained version in the near future and test it for different NLP tasks- such as topic classification, sentiment analysis, etc.

System	F1 (macro)	Accuracy
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
MC-CNN	**0.7868**	**0.8384**

(a) Sub-task A.

System	F1 (macro)	Accuracy
All TIN baseline	0.4702	0.8875
All UNT baseline	0.1011	0.1125
MC-CNN	**0.7386**	**0.9042**

(b) Sub-task B.

System	F1 (macro)	Accuracy
All GRP baseline	0.1787	0.3662
All IND baseline	0.2130	0.4695
All OTH baseline	0.0941	0.1643
MC-CNN	**0.5909**	**0.7042**

(c) Sub-task C.

Table 4: F_1Score And Accuracy Of MC-CNN Comparing To Baselines At OffensEval.

References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics", Minneapolis, Minnesota.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Alon Rozental, Daniel Fleischer, and Zohar Kelrich. 2018. Amobee at iest 2018: Transfer learning from language models. *arXiv preprint arXiv:1808.08782*.

Alon Rozental, Zohar Kelrich, and Daniel Fleischer. 2019. Latent universal task specific bert.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification.

CIC at SemEval-2019 Task 5: Simple Yet Very Efficient Approach to Hate Speech Detection, Aggressive Behavior Detection, and Target Classification in Twitter

Iqra Ameer, Muhammad Hammad Fahim Siddiqui, Grigori Sidorov, and Alexander Gelbukh
Instituto Politécnico Nacional (IPN),
Center for Computing Research (CIC),
Av. Juan Dios Batiz, s/n, Zacatenco, 07738, Mexico City,
Mexico
{iqraameer133,hammad.fahim57}@gmail.com, {sidorov,gelbukh}@cic.ipn.mx

Abstract

In recent years, the use of social media has increased incredibly. Social media permits Inter-net users a friendly platform to express their views and opinions. Along with these nice and distinct communication chances, it also allows bad things like usage of hate speech. Online automatic hate speech detection in various aspects is a significant scientific problem. This paper presents the Instituto Politécnico Nacional (Mexico) approach for the Semeval 2019 Task-5 [Hateval 2019] (Basile et al., 2019) competition for Multilingual Detection of Hate Speech on Twitter. The goal of this paper is to detect (A) Hate speech against immigrants and women, (B) Aggressive behavior and target classification, both for English and Spanish. In the proposed approach, we used a bag of words model with preprocessing (stemming and stop words removal). We submitted two different systems with names: (i) CIC-1 and (ii) CIC-2 for Hateval 2019 shared task. We used TF values in the first system and TF-IDF for the second system. The first system, CIC-1 got 2nd rank in subtask B for both English and Spanish languages with EMR score of 0.568 for English and 0.675 for Spanish. The second system, CIC-2 was ranked 4th in subtask A and 1st in subtask B for Spanish language with a macro-F1 score of 0.727 and EMR score of 0.705 respectively.

1 Introduction

The social media applications enable users to discover, create and share contents handily, without specific expertise. This remarkably boosted the amount of data generated by the users, within a process that some people call "democratization" of the web (Silva et al., 2016). Still, this liberty also permits for the publication of data, which is insulting and hurtful both regarding the ethics of democracy and the privileges of some categories of people – hate speech (HS). The Hate Speech (HS) term is defined in the literature as an expression *"that is abusive, insulting, intimidating, harassing, and incites to violence, hatred, or discrimination. It is directed against people by their race, ethnic origin, religion, gender, age, physical condition, disability, sexual orientation, political conviction, and so forth."* (Erjavec and Kovacic, 2012). HS has turned into the main issue for each sort of online website, where user-produced content comes into sight: from the comments on any post to live chatting in online games. Such material can isolate users and inflame violence (Allan, 2013). Website operators as Facebook, Twitter, and gaming companies like Runic Games recognize that hateful data are creating both practical and ethical issues and have attempted to demoralize them, causing changes in their platforms or strategies.

As stated by Pew[1], women experienced more sexualized forms of abuse than men. Platforms as Twitter are flopping in acting immediately against real-time misogyny and taking a lot of time to delete the hateful data[2]. The researchers began to concentrate on this problem and are building techniques to detect misogyny in real time (Fersini et al., 2018; Hewitt et al., 2016; Poland, 2016). Real-time HS about groups of people like asylum searchers and visitors is common all over the world, but it is rarely investigated.

[1] http://www.pewinternet.org/2017/07/11/online-harassment-2017/ Last visited: 01/02/2019

[2] https://www.telegraph.co.uk/news/2017/08/21/twitter-failing-women-taking-long-remove-misogynistic-abuse/ Last visited: 01/02/2019

In this article, we worked on the detection of (A) Hate speech against immigrants and women, (B) Aggressive behavior and target classification, both for English and Spanish languages at Hateval 2019. For this task, we submitted two systems with names: (i) CIC-1 and (ii) CIC-2. We used the bag-of-words model (plus stemming) with TF and TF-IDF as feature values and then we classified these vectors using various machine learning classifiers. We submitted two approaches (systems). Subtask A is ranked by macro-F1 score, whereas subtask B is ranked by EMR score. Our system CIC-1 got 2nd rank in subtask B for the both English (2nd out of 42 teams) and Spanish (2nd out of 25 teams) languages with EMR score of 0.568 for English; 0.675 for Spanish (accuracy score of 0.766 for English; 0.787 for Spanish). The second system, CIC-2 was ranked 4th (out of 39 teams) in subtask A and 1st (out of 23 teams) in subtask B for Spanish language with a macro-F1 score of 0.727 and EMR score of 0.705 respectively (accuracy score of 0.727 in subtask A; 0.791 for subtask B).

2 Related work

A wide range of work has been devoted to HS detection. Xu et al. (2012) applied sentiment analysis to classify bullying in tweets with the usage of Latent Dirichlet Allocation (LDA) topic models (Blei et al., 2003) to recognize related topics in these scripts.

HS detection has been improved by a diverse range of features such as n-grams (Nobata et al., 2016), character n-grams (Mehdad and Tetreault, 2016), paragraph embeddings (Nobata et al., 2016; Djuric et al., 2015) and average word embeddings. (Silva et al., 2016) proposed to detect target groups regarding their class and background on Twitter by looking for sentence structures like "I <intensity> hate <targeted group>".

Currently, interest is increasing in the identification of HS against women on the web (Ging et al., 2018). Initially, Hewitt (2016) worked on the identification of HS against women in social media. Fox (2015) observed that the reaction on hated contents posted against women by unknown and know accounts is different. In (Fox et al., 2015), the authors study the roles of anonymity and interactivity in response to sexist content posted on social media. They inferred that content from unknown account advances more prominent threatening sexism than the known ones.

3 Corpora and task description

Multilingual detection of hate speech on Twitter shared task at Hateval 2019 had two datasets for the English and Spanish language. We participated in both subtasks for both languages.

3.1 Corpora

Corpora for the training of the model consist of 9,000 labeled tweets, and the development dataset includes 1,000 unlabeled tweets. The English data set statistics of different labels is given in Table 1 and the Spanish statistics in Table 2. The corpora is manually labeled by different annotators according to three types:

- Hate speech (present vs not-present),
- Target range (whole group vs individual),
- Aggressiveness (present vs not-present).

We describe these types in the following section.

Type	Labels	Train	Dev
Hate Speech	Present (0)	2631	278
	Absent (1)	1838	222
Target Range	Whole group (0)	3352	363
	Individual (1)	1117	137
Aggres-siveness	Present (0)	2984	324
	Absent (1)	1485	176

Table 1: Spanish dataset statistics.

Type	Labels	Train	Dev
Hate Speech	Present (0)	3783	427
	Absent (1)	5217	573
Target Range	Whole group (0)	7659	781
	Individual (1)	1341	219
Aggres-siveness	Present (0)	1559	204
	Absent (1)	7441	796

Table 2: English dataset statistics.

3.2 Description of the subtasks

Subtask A: Hate speech detection against immigrants and women: it is a binary classification problem, where it is asked to predict if a specific piece of text (tweet) with a given target (women or immigrants) expresses hatred or not. The systems are evaluated using standard evaluation metrics, containing accuracy, precision, recall, and macro-F1 score. The submissions are ranked by macro-F1 score.

Subtask B: aggressive behavior and target classification: it is required to identify hatred text (tweet) (e.g., tweets, where there is HS against

Team	Task	Classifier	English	Rank$_{eng.}$	Spanish	Rank$_{spa.}$
CIC-1	A_{F1}	Logistic Regression	0.462	29 of 69	0.703	18 of 39
	B_{EMR}	Majority Voting	0.568	**2 of 41**	0.675	**2 of 23**
CIC-2	A_{F1}	MultinomialNB	0.494	14 of 69	0.727	**4 of 39**
	B_{EMR}	Classifiers Chain	0.314	19 of 41	0.705	**1 of 23**

Table 3: Our results of Hateval 2019 shared task with ranking for both subtasks A and B.

women or immigrants were marked before) as aggressive or not, and on the second place to recognize a harassing target, either the text (tweet) is against an individual or a group. The evaluation of subtask B was carried out using a partial match and exact match (Basile et al., 2019). The submissions are ranked by EMR score. A tweet must be identified exclusively in one of the following types:

1. **Hateful:** an expression with feelings of dislike, very unpleasant or filled with hatred.
2. **Target Range:** the tweet contains offensive messages intentionally sent to a particular individual or to a group.
3. **Aggressiveness:** it is based on the person's purpose to be aggressive, damaging, or even to provoke.

3.3 Baselines

The Hateval 2019 has set up two following baselines:

- **SVC baseline:** the SVC baseline is a linear Support Vector Machine (SVM) based on TF-IDF representation.
- **MFC baseline:** The MFC baseline is a trivial model that assigns the most frequent label (estimated on the training set) to all the instances in the test set.

4 Description of our approach

In this section, we describe the two submitted approaches (systems) considering the features and machines learning models used for this shared task.

4.1 Pre-processing

We performed pre-processing on raw tweets before feature extraction. Pre-processing helps in these kind of tasks (Markov et al., 2017). For both approaches we used stemming and stop words removal. In CIC-2 we additionally made the following steps:

- we removed HTML tags,
- punctuation marks are removed,
- special characters are removed, like "&", "$", "_", ", ", etc.

4.2 Features

The pre-processed text was used to generate the features for the machine learning (ML) algorithms. We used a well-known bag of words model, for example, (Sidorov, 2013; Sidorov, 2019). For the first system, we used TF and for the second system TF-IDF values.

4.3 Machine learning algorithms

In our two systems, we used four different classifiers for both subtasks A and B. In CIC-1: Subtask A: Logistic regression, subtask B: Majority voting. In CIC-2: Subtask A: Multinomial Naive Bayes, subtask B: Classifier chains. For all classifiers, we used available implementation in scikit-learn[3].

5 Results and analysis

Results of our both systems CIC-1 and CIC-2 are presented in Table 3, for both shared subtasks, i.e., A and B with our rank in Hateval 2019 competition. Table 3, subtask A ranked by macro-F1 and B by EMR, we used the following conventions. In the first column, "Team" refers to both different systems (CIC-1 and CIC-2) submitted for the shared task. "Task" represents two different subtasks A and B (A_{F1} means that scores of the subtask A are ranked by macro-F1 and B_{EMR} means that scores of the subtask B are ranked by EMR), see section 3.2. "Classifier" states different classifiers, which we used in this competition. "English" and "Spanish" indicate scores for English and Spanish respectively. "Rank$_{eng.}$" and "Rank$_{spa.}$" mean our team's rank in the competition in both subtasks.

The system CIC-1 got 2nd rank in subtask B for the both English and Spanish languages with EMR score of 0.568 for English; 0.675 for Spanish. We used majority voting classifier for both languages.

[3] https://scikit-learn.org/sFigure/ Last visited: 01/04/2019

User name	Macro-F1	Acc.	Rank
saradhix	0.651	0.653	1
Panaetius	0.571	0.572	2
YunxiaDing	0.546	0.560	3
alonzorz	0.535	0.558	4
amontejo	0.519	0.535	5
:	:	:	:
:	:	:	:
hammad.fahim57	**0.494**	0.523	**14**
:	:	:	:
:	:	:	:
Iqraameer133	**0.462**	0.505	**29...**
SVC baseline	0.451	0.492	-
MFC baseline	0.367	0.579	-

Table 4: Results for Subtask A - English.

User name	EMR	Acc.	Rank
MFC baseline	0.580	0.802	-
ninab	0.570	0.802	1
iqraameer133	**0.568**	0.766	**2**
scmhl5	0.483	0.770	3
garain	0.482	0.763	4
gertner	0.399	0.631	5
:	:	:	:
:	:	:	:
hammad.fahim57	**0.314**	0.711	**19...**
SVC baseline	0.308	0.692	-

Table 5: Results for Subtask B - English.

User name	Macro-F1	Acc.	Rank
francolq2	0.730	0.731	1
luiso.vega	0.730	0.734	2
gertner	0.729	0.729	3
hammad.fahim57	**0.727**	0.758	**4**
dibesa	0.725	0.728	5
:	:	:	:
:	:	:	:
Iqraameer133	**0.703**	0.708	**18...**
SVC baseline	0.701	0.705	-
MFC baseline	0.370	0.588	-

Table 6: Top 5 teams for Subtask A - Spanish.

User name	EMR	Acc.	Rank
SVC baseline (as by organizers)	0.771	0.771	-
hammad.fahim57	**0.705**	**0.791**	**1**
MFC baseline	0.704	0.704	-
iqraameer133	**0.675**	0.787	**2**
gertner	0.671	0.758	3
francolq2	0.657	0.749	4
OscarGaribo	0.644	0.732	5...
SVC baseline(Our)		0.550	

Table 7: Top 5 teams for subtask B - Spanish

The system CIC-2 CIC-2 ranked 4th in subtask A and 1st in subtask B for Spanish language with a macro-F1 score of 0.727 and EMR score of 0.705 respectively by using MultinomialNB classifier. It is clear that our system was able to get good results in subtask B (to classify aggressive behavior and target), but was not able to perform well in subtask A (to detect hate speech against immigrants and women) for English language, although we obtained the 2nd position in subtask A for Spanish language. For Spanish subtask B, we tried to reproduce SVM baseline as by organizers but we failed, our SVM baseline gave us 0.550 accuracy.

We made experiments without stop words removal and stemming, and accuracy, in this case, goes down by 2-3%. We discovered that imbalanced data was the main reason for poor performance on English for subtasks A and B. We noticed that most of the submitted systems achieved poor results on the subtask A.

6 Conclusion and future work

In this article, we described our approach to detect (1) Hate Speech Detection against immigrants and women; (2) aggressive behavior and target on the Twitter corpus. We submitted two different systems namely: (i) CIC-1 and (i) CIC-2. We used a bag of words model with TF and TF-IDF values. The vectors are then used as features for classifiers like MultinomialNB, Majority Voting, Logistic Regression, and Classifier Chains. Our CIC-1 system ranked 2nd in task B for both English and Spanish languages. Our system CIC-2 ranked 1st in task B for Spanish and 4th for the same language in task A.

In future work, we can consider embeddings with TF-IDF weighting (Arroyo-Fernández et al., 2019) and learning of document embeddings like in (Gómez-Adorno et al,. 2018). We also plan to consider syntactic n-grams (n-grams obtained by following paths in syntactic dependency trees) (Sidorov 2013; 2019).

We have also made the winning model public[4] for other researchers to use.

Acknowledgement

The is work done with partial support of Mexican government (Conacyt, SNI, COFAA-IPN). We also acknowledge the support of Conacyt project 240844 and SIP-IPN projects 20171813, 20181849, and 20195719.

[4] https://github.com/iqraameer133/HateEval2019 Last visited 13/02/2019

References

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. Learning from bullying traces in social media. In *Proceedings of the 2012 Conference of North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT),* Montreal, Canada, 2012, pp.656-666.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*, pages 145–153, Geneva, Switzerland.

David M. Blei, Andrew Y. Ng and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3(Jan):993–1022.

Debbie Ging and Eugenia Siapera (eds.). 2018. Special issue on online misogyny. *Feminist Media Studies*, pages 515–524.

Elisabetta Fersini, Debora Nozza, and Paolo Rosso. 2018. Overview of the evalita 2018 task on automatic misogyny identification (ami). In *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18),* Turin, Italy. CEUR.org.

Grigori Sidorov. *Syntactic N-grams in Computational Linguistics*. Springer, 2019, 125 p.

Grigori Sidorov. 2013. *Construcción no lineal de n-gramas en la lingüística computacional* [*Non-linear Construction of N-grams in Computational Linguistics*]. Sociedad Mexicana de Inteligencia Artificial, 166 p.

Helena Gómez-Adorno, Juan-Pablo Posadas-Durán, Grigori Sidorov and David Pinto. 2018. Document embeddings learned on various types of n-grams for cross-topic authorship attribution. *Computing*, pages 1–16.

Ilia Markov, Efstathios Stamatatos and Grigori Sidorov. 2017. Improving Cross-Topic Authorship Attribution: The Role of Pre-Processing. In *Proceedings of the 18th International Conference on Computational Linguistics and Intelligent Text Processing* (*CICLing 2017*), Budapest, Hungary. Springer.

Ignacio Arroyo-Fernández, Carlos-Francisco Méndez-Cruz, Gerardo Sierra, Juan-Manuel Torres-Moreno and Grigori Sidorov. 2019. Unsupervised Sentence Representations as Word Information Series: Revisiting TF–IDF. *Computer Speech & Language*, 10.1016/j.csl.2019.01.005.

Jesse Fox, Carlos Cruz, and Ji Young Lee. 2015. Perpetuating online sexism offline: Anonymity, interactivity, and the effects of sexist hashtags on social media. *Computers in Human Behavior,* pages 436–442.

Leandro Silva, Mainack Mondal, Denzil Correa, Fabricio Benevenuto and Ingmar Weber. 2016. Analyzing the Targets of Hate in Online Social Media. In *Proceedings of the 10th International AAAI Conference on Web and Social Media* (*ICWSM'16*)*,* pages 687–690.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic and Narayan Bhamidipati. 2015. Hate Speech Detection with Comment Embeddings. In *Proceedings of the 24th International Conference on World Wide Web*.

Sarah Hewitt, Thanassis Tiropanis and Christian Bokhove. 2016. The problem of identifying misogynist language on twitter (and other online social spaces). In *Proceedings of the 8th ACM Conference on Web Science* (*WebSci'16*)*,* pages 333–335, ACM.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Rangel Francisco, Paolo Rosso and Manuela Sanguinetti. 2019. SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter. In Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), *Association for Computational Linguistics,* Minneapolis, Minnesota, USA.

Yashar Mehdad and Joel Tetreault. 2016. Do characters abuse more than words? In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue,* pages 299–303, Los Angeles, CA, USA.

CiTIUS-COLE at SemEval-2019 Task 5: Combining Linguistic Features to Identify Hate Speech Against Immigrants and Women on Multilingual Tweets

Sattam Almatarneh
(CiTIUS)
Universidade de Santiago de Compostela, Spain
University of Vigo, Spain
sattam.almatarneh@usc.es

Pablo Gamallo
(CiTIUS)
Universidade de Santiago de Compostela, Spain
pablo.gamallo@usc.es

Francisco J. Ribadas Pena
Department of Computer Science
University of Vigo, Spain
ribadas@uvigo.es

Abstract

This article describes the strategy submitted by the CiTIUS-COLE team to SemEval 2019 Task 5, a task which consists of binary classification where the system predicts whether a tweet in English or in Spanish is hateful against women or immigrants or not. The proposed strategy relies on combining linguistic features to improve the classifier's performance. More precisely, the method combines textual and lexical features, embedding words with the bag of words in Term Frequency-Inverse Document Frequency (TF-IDF) representation. The system performance reaches about 81% F1 when it is applied to the training dataset, but its F1 drops to 36% on the official test dataset for the English and 64% for the Spanish language concerning the hate speech class.

1 Introduction

Hate speech is usually defined as any communication that derogates a person or a group based on some characteristic such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or another characteristic (Schmidt and Wiegand, 2017). The spread of the Internet and the increasing use of social networks has led people to have an increased willingness to express their opinions online. Despite the great benefits of using the Internet and particularly the social networks, the risk is that people are more likely to adopt aggressive behavior because of the anonymity provided by these environments. This contributes to the propagation of hate speech as well. Since this type of tendentious communication can be extremely harmful to society, governments and social network platforms can benefit from detection and prevention tools. The scientific study of hate speech, from a computer science point of view, is recent, and the number of studies in the field is low (Fortuna and Nunes, 2018). The goal of SemEval-2019 Task 5 as described in Basile et al. (2019) is Hate Speech detection in Twitter focused on two specific targets, women and immigrants. The task is organized in two related sub-tasks for each language (English and Spanish):

- TASK A - Hate Speech Detection against Immigrants and Women
- TASK B - Aggressive behavior and Target Classification.

In this article, we describe our proposed system for task A only. Our approach is mainly based on the generation of corpus-based dictionaries containing hate speech words which are used in addition to other linguistic features to improve the efficiency in detecting hate speech in both English and Spanish languages.

This paper is organized as follows. The method is described in Section 2. Experiments, results, and a discussion on them are presented in Section 3. Finally, conclusions are addressed in Section 4.

2 Method

We deal with the task by automatic classifiers composed of training data in a supervised strategy. The characteristics of tweets are encoded as features in vector representation. These vectors and the corresponding labels feed the classifiers.

2.1 Features

Linguistic features are the most important and influential factor in increasing the efficiency of classifiers for any task of text mining. Many studies examined the impact of these features in many tasks such as polarity classification (Almatarneh and Gamallo, 2018b, 2019). In this study, we included a number of linguistic features for the task of determining hate speech in tweets. The main linguistic features we will use and analyze are the following: N-grams, word embeddings, and lexical features.

2.1.1 TF-IDF features

We model texts by n-grams based on the occurrence of unigrams of words that occur in documents. The unigrams are very valuable elements to find very relevant expressions in the domain of interest. All terms are assigned a weight by TF-IDF which is computed in Equation 1:

$$tf/idf_{t,d} = (1 + log(tf_{t,d})) \times log(\frac{N}{df_t}), \quad (1)$$

where $tf_{t,d}$ is the term frequency of term t in document d. N stands for the the number of documents in the collection and, df_t represents the number of documents in the collection containing t. To transform the tweets into a matrix of TF-IDF features, we used *sklearn* feature extraction Python library.[1]

2.1.2 Doc2Vec

To represent the tweets, we make use of the *Doc2Vec* algorithm described in Le and Mikolov (2014). This neural-based model is efficient when you have to account for high-dimensional and sparse data (Le and Mikolov, 2014; Dai et al., 2015). Doc2vec learns corpus features using an unsupervised strategy and provides a fixed-length feature vector as output. The output is then fed into a machine learning classifier. We used a freely available implementation of the Doc2Vec algorithm included in gensim, [2] which is a free Python library. The implementation of the Doc2Vec algorithm requires the number of features to be returned (length of the vector). Thus, we performed a grid search over the fixed vector length 100 (Collobert et al., 2011; Mikolov et al., 2013a,b).

2.1.3 Lexical features

Lexical features consist of specific words identified as belonging to the class of hate speech. For instance, as word *bitch* can be associated with hate speech, it will be added to a specific dictionary containing words associated with hate speech. In addition, a weight is assigned to each word. The higher the weight the more intense the hate value of the word. We automatically built several weighted dictionaries from the annotated corpus:

- Dictionary of lexical words 295 English words and 262 Spanish words.
- Dictionary of hashtags: 1090 English hashtags and 201 Spanish hashtags.
- Dictionary of address references: 1661 English references and 1263 Spanish references.

We just considered words belonging to lexical categories, hence, only nouns, verbs, adjectives, and adverbs were selected. PoS tagging for English and Spanish was carried out with the multilingual toolkit LinguaKit (Gamallo et al., 2018).

The method to build the hate speech dictionaries is somehow inspired by that reported in Almatarneh and Gamallo (2018a, 2017) for very negative opinions. The hate speech score of a word, noted HS, is computed as follows:

$$HS(w) = \frac{freq_{total}(w)}{freq_{hs}(w)} \quad (2)$$

where $freq_{total}(w)$ is the number of occurrences of word w in the whole corpus, and $freq_{hs}(w)$ stands for the number of occurrences of the same word in the segments (tweets) annotated as hate speech. In addition to the hate speech score HS, it is also required to compute a threshold above which the word is considered hate speech. So, we compute the difference between the use of a word as hate speech and as not:

$$DIFF(w) = freq_{hs}(w) - freq_{-hs}(w) \quad (3)$$

where $freq_{-hs}(w)$ stands for the occurrences of w in segments that are not hate speech. To insert a word in the dictionary, the value of $DIFF(w)$ must be higher than a experimentally set threshold. In our experiments, this value was 5. So, in our dictionaries, we only selected those words (hashtags or references) with $DIFF$ values higher than 5. Finally, words were ranked by their HS score giving rise to weighted and ranked lexicons.

3 Experiments

The main datasets that were used for training and testing our model are described in Basile et al. (2019). This article describes the SemEval-2019 Shared Task 5 aimed at Hate Speech detection in Twitter.

[1] http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html#sklearn.feature_extraction.text.TfidfVectorizer

[2] https://radimrehurek.com/gensim/

Table 1: Performance on the training dataset with different feature configurations of TF-IDF, Doc2Vec and lexicons for English language.

Featuers	Hate			Not			Avg F1	Accuracy
	Prec.	Rec.	F1	Prec.	Rec.	F1		
TF-IDF	0.72	0.68	0.70	0.79	0.82	0.80	0.76	0.76
Doc2Vec	0.68	0.52	0.59	0.71	0.83	0.77	0.69	0.70
Lexicon	0.69	0.46	0.55	0.7	0.86	0.77	0.68	0.69
Doc2Vec + Lexicon	0.71	0.57	0.63	0.74	0.84	0.79	0.72	0.73
All	0.78	0.74	**0.76**	0.83	0.86	**0.84**	**0.81**	**0.81**

Table 2: Performance on the training dataset with different feature configurations of TF-IDF, Doc2Vec and lexicons for Spanish.

Featuers	Hate			Not			Avg F1	Accuracy
	Prec.	Rec.	F1	Prec.	Rec.	F1		
TF-IDF	0.75	0.7	0.72	0.78	0.83	0.80	0.77	0.77
Doc2Vec	0.57	0.11	0.18	0.58	0.94	0.72	0.49	0.58
Lexicon	0.82	0.66	0.73	0.78	0.89	0.83	0.79	0.79
Doc2Vec + Lexicon	0.82	0.68	0.74	0.78	0.89	0.83	0.79	0.80
TF-IDF + Lexicon	0.81	0.76	**0.78**	0.83	0.86	**0.85**	**0.82**	**0.82**

3.1 Development and training

As we considered that the size of the training collection provided by the organizers was not large enough, we made use of another available training data for the same task to build our lexicons.[3] The algorithm to build the lexicons has been described above in Subsection 2.1.3.

As far as the classification strategy is concerned, we decided to use *sklearn.svm.LinearSVC* for learning the classifiers.[4] Suport Vector Machine (SVM) proved to be the best strategy for detecting extreme opinions in previous work (Almatarneh and Gamallo, 2019)

The training dataset provided by the organizers of the shared task was used as a development corpus so as to learn the best feature configuration using 10-fold cross-validation.

Tables 1 and 2 shows the result of the experiments on the training corpus. In these tables, we depict the performance of all tested features in both English and Spanish Languages. The combination of all features (TF-IDF, Doc2Vec, and lexicons) gives the best performance for English. However, in Spanish the use of Doc2Vec made it lower the performance as the best F1 was achieved by just combining TF-IDF with lexical features.

3.2 Test

Taking into account the results shown in tables 1 and 2, we submitted two different model configurations for English and Spanish testing. More precisely, for English TASK A we used the combination of all features, whereas the Spanish model in TASK A was built by only combining lexical and TF-IDF features.

Unlike the experiments on the training dataset, our approach showed bad performance on the test dataset as Table 3 shows.

The poor scores in the English dataset are due to the strange behavior of our approach with the non-hate speech class of speech class. Recall on this class was merely 0.07 while on the target class reached 0.97.

4 Conclusions and Future Work

The approach we developed for the task of hate speech detection in English and Spanish is mainly based on the generation of lexicons containing hate speech words. Lexicons are used in addition to other linguistic features (TF-IDF and Doc2Vec) to improve the efficiency of a SVM classifier.

[3] https://github.com/ZeerakW/hatespeech/blob/master/NAACL_SRW_2016.csv

[4] https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

Table 3: Performance of our approach on the test dataset for English and Spanish languages

Featuers	Hate			Not			Avg F1	Accuracy
	Prec.	Rec.	F1	Prec.	Rec.	F1		
TASK A English	0.43	0.97	0.60	0.77	0.07	0.13	0.36	0.45
TASK A Spanish	0.59	0.54	0.56	0.70	0.74	0.72	0.64	0.66

Even if we obtained acceptable results in the development phase using the training corpus (more than 0.80 F1 score), the results achieved in the test phase were disappointing, especially for the English language.

In order to discover the problems underlying our overfitted model, a deep error analysis will be performed. Once released the dataset test, we will be able to analyze the contribution of each of the features used so that we can check if it was one of the lexicons that caused the low performance of our system.

In future work, our objective is to improve the basic method to build hate speech lexicons (and related topics) from annotated corpora in order to use them in both supervised and unsupervised strategies.

Acknowledgments

This work has received financial support from project TelePares (MINECO, ref:FFI2014-51978-C2-1-R), and the Consellera de Cultura, Educacin e Ordenacin Universitaria (accreditation 2016-2019, ED431G/08) and the European Regional Development Fund (ERDF).

References

Sattam Almatarneh and Pablo Gamallo. 2017. Automatic construction of domain-specific sentiment lexicons for polarity classification. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 175–182. Springer.

Sattam Almatarneh and Pablo Gamallo. 2018a. A lexicon based method to search for extreme opinions. *PloS one*, 13(5):e0197816.

Sattam Almatarneh and Pablo Gamallo. 2018b. Linguistic features to identify extreme opinions: An empirical study. In *Intelligent Data Engineering and Automated Learning – IDEAL 2018*, pages 215–223, Cham. Springer International Publishing.

Sattam Almatarneh and Pablo Gamallo. 2019. Comparing supervised machine learning strategies and linguistic features to search for very negative opinions. *Information*, 10(1).

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics, location = Minneapolis, Minnesota.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

Andrew M Dai, Christopher Olah, and Quoc V Le. 2015. Document embedding with paragraph vectors. *arXiv preprint arXiv:1507.07998*.

Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Comput. Surv.*, 51(4):85:1–85:30.

P. Gamallo, M. Garcia, C. Pieiro, R. Martinez-Castao, and J. C. Pichel. 2018. Linguakit: A big data-based multilingual tool for linguistic analysis and information extraction. In *2018 Fifth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 239–244.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Grunn2019 at SemEval-2019 Task 5: Shared Task on Multilingual Detection of Hate

Mike Zhang, Roy David, Leon Graumans, Gerben Timmerman
University of Groningen
Groningen, the Netherlands
{j.j.zhang.1, r.a.david, l.r.n.graumans, g.h.r.timmerman}@student.rug.nl

Abstract

Hate speech occurs more often than ever and polarizes society. To help counter this polarization, SemEval 2019 organizes a shared task called the Multilingual Detection of Hate. The first task (A) is to decide whether a given tweet contains hate against immigrants or women, in a multilingual perspective, for English and Spanish. In the second task (B), the system is also asked to classify the following subtasks: hateful tweets as aggressive or not aggressive, and to identify the target harassed as individual or generic. We evaluate multiple models, and finally combine them in an ensemble setting. This ensemble setting is built of five and three submodels for the English and Spanish task respectively. In the current setup it shows that using a bigger ensemble for English tweets performs mediocre, while a slightly smaller ensemble does work well for detecting hate speech in Spanish tweets. Our results on the test set for English show 0.378 macro F1 on task A and 0.553 macro F1 on task B. For Spanish the results are significantly higher, 0.701 macro F1 on task A and 0.734 macro F1 for task B.

1 Introduction

The increasing popularity of social media platforms such as Twitter for both personal and political communication has seen a well-acknowledged rise in the presence of toxic and abusive speech on these platforms (Kshirsagar et al., 2018). Although the terms of services on these platforms typically forbid hateful and harassing speech, the volume of data requires that ways are found to classify online content automatically. The problem of detecting, and therefore possibly limit the hate speech diffusion, is becoming fundamental (Nobata et al., 2016).

Previous work concerning hate speech against immigrants and women such as Olteanu et al. (2018) observed that extremist violence tends to lead to an increase in online hate speech, particularly on messages directly advocating violence. Also, Anzovino et al. (2018) contributed to the research field by (1) making a corpus of misogynous tweets, labelled from different perspective and (2) created an exploratory investigations on NLP features and ML models for detecting and classifying misogynistic language.

Basile et al. (2019) proposed a shared task on the Multilingual Detection of Hate, where participants have to detect hate speech against immigrants and women in Twitter, in a multilingual perspective, for English and Spanish. The task is divided in two related subtasks for both languages: a basic task about hate speech, and another one where fine-grained features of hateful contents will be investigated in order to understand how existing approaches may deal with the identification of especially dangerous forms of hate, for example those where the incitement is against an individual rather than against a group of people, and where an aggressive behavior of the author can be identified as a prominent feature of the expression of hate.

Within this experiment, Task A is a binary classification task where our system has to predict whether a tweet is hateful or not hateful. For Task B, our system has to decide whether a tweet is aggressive or not aggressive, and whether that tweet targets an individual or generic group, to elaborate, a single human or group of people.

The paper is structures as follows. In section 2 our system setup is described. In section 3, the datasets together with the preprocessing steps are presented. In section 4, obtained results are detailed. Finally, in section 5 a discussion about the proposed system is outlined.

2 System Setup

In our approach, we trained multiple classifiers and combined their results into an ensemble model using majority vote.

English Ensemble Setup

The setup of our system optimized for the English classification tasks consisted of the following classifiers:

- Random Forest
- Support Vector Machine (1)
- Support Vector Machine (2)
- Logistic Regression
- BiLSTM

Spanish Ensemble Setup

Due to time restrictions, we used three classifiers for the Spanish tasks:

- Random Forest
- Support Vector Machine (1)
- Logistic Regression

These time restrictions occurred, because we decided in the last moment to run our system for the Spanish task too. However, we did not have hate speech specific word embeddings, nor trained a BiLSTM model for the Spanish task. Therefore, we decided to run only three classifiers for both the Spanish tasks.

2.1 Random Forest (RF)

For our RF model we executed a grid search starting with the following parameters: character n-grams with range: 2-3, 2-4, 1-3, 1-4; word n-grams with range 1, 1-2, 1-3, 1-4 and all combinations of them. In the end we used a tf-idf vectorizer with character n-grams with range 2-4. As for our parameters also following a grid search we used 400 estimators, entropy as our split criterion/estimator, balanced for our class weight and a random seed of 1337. Due to time restrictions, we used the same parameters for the Spanish tasks.

2.2 Support Vector Machine (SVM 1)

Within this subpart of our ensemble model, we used a SVM model from the scikit-learn library (Pedregosa et al., 2011). We used a *linear* `kernel`, and a *weighted* `class_weight`. This model used vectorized character n-grams in range 2-4 using a tf-idf vectorizer as its input.

2.3 Support Vector Machine (SVM 2)

We used a second SVM classifier within our ensemble model, but this time with word embeddings as its input. This choice is motivated by the hypothesis that introducing different predictions given by models trained differently could lead to more insights. We tested four pre-trained embedding representations, which are the following: the 300-dimensional `GloVe` embeddings and the 25-dimensional `GloVe Twitter` representations by Pennington et al. (2014); a 400-dimensional and 100-dimensional word embedding created from tweets (Van der Goot). Using the `GloVe` embeddings proved to be superior within our work. The results of each word embedding can be found in Table 6.

2.4 Logistic Regression (LR)

Following our grid search testing our LR model with a tf-idf vectorizer with the following parameters: character n-grams with range: 2-3, 2-4, 1-3, 1-4; word n-grams with range 1, 1-2, 1-3, 1-4 and all combinations of them, we got the best performance using a tf-idf vectorizer with character n-grams with range 2-4. Due to time restrictions, we used the same parameters for the Spanish tasks.

2.5 BiLSTM

Our BiLSTM classifier was only optimized for the English classification task. Hence we decided not to use it in our Spanish setup. In combination with the BiLSTM model we used an attention mechanism, as proposed by Yang et al. (2016).

LSTM models can handle input sequentially and therefore can take word order into account. We combine this with a bidirectional model, which allows us to process the tweets both forwards and backwards. For each word in the tweets, the LSTM model combines its previous hidden state and the current word's embedding weight to compute a new hidden state. After using dropout to shut down a percentage of neurons of the model, we feed the information to the at-

	Hate Speech		Target Range		Aggressiveness	
	0	1	0	1	0	1
Trial data	50	50	87	13	80	20
Train data	5217	3783	7659	1341	7440	1559
Dev data	573	427	781	219	796	204
Test data	3000					
Total	13100					

Table 1: Distribution of English data, labels of the test data were not specified.

	Hate Speech		Target Range		Aggressiveness	
	0	1	0	1	0	1
Train data	2643	1857	3371	1129	2998	1502
Dev data	278	222	363	137	324	176
Test data	1600					
Total	6600					

Table 2: Distribution of Spanish data. No trial data was available, and test data labels were not specified.

tention mechanism. This mechanism emphasizes the most informative words in the article and gives these more weight.

Our final model uses 512 units in the hidden layer of the BiLSTM, a batch size of 64, the Adam optimizer in combination with the default learning rate of 0.001 and a dropout of 0.4. We trained our model for 10 epochs, of which we saved the model with the lowest validation loss.

3 Data and Preprocessing

For this shared task, the data distribution is seen in Table 1 and Table 2 for the train and development data, we assumed the trial data to be train data too. After release of the test data, the distribution would be 69% train, 8% development, and 23% (3000 sentences) test data for the English task, and for the Spanish task 68% train, 8% development, and 24% (1600 sentences) test data. For final submission, we combined the train and development data to train our system on.

The meaning of the binary encoding is as follows, for Hate Speech (HS) and Aggressiveness (AG): 0 or 1, absent and present respectively. For Target Range (TR): 0 or 1, whole group and individual respectively. We notice that there is more data available for the English task than the Spanish one.

With regard to preprocessing, we did this in the following fashion:

- Tokenized with the NLTK TweetTokenizer.
- Replaced URLs with a placeholder suitable for our English embeddings.
- Replaced mentions with a placeholder suitable for the available English embeddings (van der Goot and van Noord, 2017).
- Converted words to lowercase.
- Filtered out stopwords using the stopwords from NLTK, either English or Spanish.
- Removing single characters, excluding emoji's.

For the BiLSTM, we did not do any preprocessing. We deemed this might affect the learning curve of the system, since a BiLSTM algorithm often performs well with lots of different data. So, without preprocessing there will be less loss of information and thus a better performing system.

We tested how the preprocessing affected our scores, results are in Table 3 and Table 4. We used the train and development data available to test the preprocessing. We started using all the preprocessing, and in a cumulative way, excluded a preprocess step one by one. So in the end, we would only have tokenization left.

Interesting is that the scores of the RF and SVM 1 model are higher, for both English and Spanish data, when we exclude preprocessing steps. At the step of replacing URLs and usernames with placeholders, we expected the scores to be higher if excluded. Because if the same URL or username occurs often in the training set, and that specific URL or username is always corresponding with a hateful or non-hateful message, our system could wrongly classify a comment in the development set containing that same URL or username. The scores also increase when we exclude lowercasing or remove single characters in addition to the placeholder steps. However, if we omit either lowercasing or characters alone, the scores do not get better than if we use all preprocessing. This also explains the higher score with the LR model, but if we only disregard the character preprocessing step, the score also does not get better.

4 Results

In this section, we state our results on the test set, as well as the results of our ensemble model and individual models on the development set. Our final system for the English task consists of all five models shown in the English Ensemble Setup, each given a result being either 0 or 1, and run a majority vote on it for a final result. For the Span-

	RF	SVM 1	SVM 2	LR	BiLSTM
All	74.2	73.9	72.7	74.0	-
- URL	74.5	73.8	68.4	73.9	-
- USERNAME	75.3	74.1	68.3	73.9	-
- Lowercase	75.8	73.8	69.2	73.5	-
- Stopwords	74.2	72.4	67.9	73.4	-
- Characters	75.6	73.0	68.9	75.2	-
No preprocess (only tokenization)	75.6	73.0	68.9	75.2	77.5

Table 3: Scores with changes in preprocessing for English, scores in bold means that it was higher than using all preprocessing of the respective system.

	RF	SVM 1	LR
All	78.2	79.9	77.8
- URL	78.7	80.1	77.4
- USERNAME	78.2	78.8	77.9
- Lowercase	75.9	79.6	76.6
- Stopwords	77.3	80.8	76.7
- Characters	75.7	81.0	77.7
No preprocess (only tokenization)	75.7	81.0	77.7

Table 4: Scores with changes in preprocessing for Spanish, csores in bold means that it was higher than using all preprocessing of the respective system.

ish task, the final system contains three models, described in the Spanish Ensemble Setup.

The results on the the various tasks we participated in are listed in Table 5. For the English task, we achieved a much lower accuracy and macro f1-score than for the Spanish task. Assuming the data has been distributed fairly for both languages, it could be that the quality of the test data is lower than the train and development data.

These scores were lower in comparison to our results of the individual classifiers on the development set which are listed in table 3 and 4.

5 Discussion

We compared multiple classification algorithms and combined them into an ensemble model to get a more robust and accurate system. Initially, our system performed reasonably well on the development set, but when tested on the final test set our performance dropped a fair bit. Overall, the drop in performance was to be expected. During the final evaluation of the test set our system predicted over 80% as hate speech. Looking at the data we thought a large part of the remaining 20% could also be classified as hate speech. Also the majority class baseline (Basile et al., 2019) ranked second for accuracy, supporting our expectations.

From our results we can conclude that using a bigger ensemble model for the English tweets performs mediocre in comparison to a smaller ensemble model for detecting hate speech in Spanish tweets.

English Task A	accuracy	macro F1
Fermi	0.653	0.651
Panaetius	0.572	0.571
YNU_DYX	0.560	0.546
Grunn2019	0.459	0.378
English Task B	EMR	macro F1
MFC baseline	0.580	0.421
ninab	0.570	0.467
CIC-1	0.568	0.551
Grunn2019	0.279	0.553
Spanish Task A	accuracy	macro F1
Atalaya	0.731	0.730
mineriaUNAM	0.734	0.730
MITRE	0.729	0.729
Grunn2019	0.708	0.701
Spanish Task B	EMR	macro F1
hammad.fahim57	0.705	0.755
CIC-1	0.675	0.649
gertner	0.671	0.772
Grunn2019	0.601	0.734

Table 5: Scores of our ensemble models on both subtasks and languages during testing phase, compared to the top three systems in that subtask.

	accuracy	macro f1-socre
GloVe 300d	0.726	0.727
Glove Twitter 25d	0.680	0.675
Twitter 100d	0.716	0.711
Twitter 400d	0.719	0.717

Table 6: Scores on the English development set of the Support Vector Machine (SVM 2) classifier using different word embeddings as input.

In the future, we would like to try to improve the performance of our Spanish model, of which our development was cut short due to time restrictions. We would also like to test our models with more high quality data. It would be interesting to find out whether this helps to improve our models' performance.

6 Acknowledgements

We would like to thank the organizers of SemEval-2019[1], the organizers of SemEval 2019 Task 5 - Shared Task on Multilingual Detection of Hate[2], and the reviewers for accepting our paper. We would also like to thank R.M. Van der Goot for his guidance and help where needed.

[1]`http://alt.qcri.org/semeval2019/`

[2]`https://competitions.codalab.org/competitions/19935`

References

Maria Anzovino, Elisabetta Fersini, and Paolo Rosso. 2018. Automatic identification and classification of misogynistic language on twitter. In *International Conference on Applications of Natural Language to Information Systems*, pages 57–64. Springer.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Rob Van der Goot. http://www.let.rug.nl/rob/.

Rob van der Goot and Gertjan van Noord. 2017. MoNoise: Modeling noise using a modular normalization system. *Computational Linguistics in the Netherlands Journal*, 7:129–144.

Rohan Kshirsagar, Tyus Cukuvac, Kathleen McKeown, and Susan McGregor. 2018. Predictive embeddings for hate speech detection on twitter. *arXiv preprint arXiv:1809.10644*.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153. International World Wide Web Conferences Steering Committee.

Alexandra Olteanu, Carlos Castillo, Jeremy Boy, and Kush R Varshney. 2018. The effect of extremist violence on hateful speech online. In *Twelfth International AAAI Conference on Web and Social Media*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

GSI-UPM at SemEval-2019 Task 5: Semantic Similarity and Word Embeddings for Multilingual Detection of Hate Speech Against Immigrants and Women on Twitter

Diego Benito, Oscar Araque, and Carlos A. Iglesias
Intelligent Systems Group
Universidad Politécnica de Madrid
Madrid, Spain
Avenida Complutense, 30
{d.benito,o.araque,carlosangel.iglesias}@upm.es

Abstract

This paper describes the GSI-UPM system for SemEval-2019 Task 5, which tackles multilingual detection of hate speech on Twitter. The main contribution of the paper is the use of a method based on word embeddings and semantic similarity combined with traditional paradigms, such as n-grams, TF-IDF and POS. This combination of several features is fine-tuned through ablation tests, demonstrating the usefulness of different features. While our approach outperforms baseline classifiers on different sub-tasks, the best of our submitted runs reached the 5th position on the Spanish subtask A.

1 Introduction

Information available in social networks is the result of many interactions between users and their activity on the net. Unfortunately, hate speech and other misuses are proliferating on the Internet. Hate speech authors justify their conduct based on the freedom of speech argument. Thus, a debate over hate speech legislation and freedom of speech has been generated (Herz and Molnar, 2012).

The task to decide if a piece of text contains hate speech is not trivial, even for humans. Being subject to different interpretations and opinions, the manifestations of hate speech become difficult to define. Based on previous hate speech statements (Fortuna and Nunes, 2018; Schmidt and Wiegand, 2017), this phenomenon could be defined as offensive or humorist content in form of text, video, or images that attacks, diminishes, incites violence or hate against groups or individuals, based on actual or perceived specific characteristics such as physical appearance, religion, descent, national or ethnic origin, sexual orientation, gender identity, or any other.

Hate speech topic has gained impact and popularity in recent years, which is reflected not only by the increased media coverage but also by the growing political attention. Regarding the specific forms of hate speech that we deal with, sexism and racism victims increased during 2017 according to the FBI hate crime statistics[1].

For this reason, participating in SemEval2019 Task 5 (Basile et al., 2019) is such an interesting challenge. The proposed task consists in Hate Speech detection in Twitter messages featured by two specific different targets intrinsically related to the phenomena mentioned above, immigrants and women. The task is enriched by adding a multilingual perspective fostering the research for both English and Spanish messages.

The system proposed relies on a supervised classifier using different text features combined with several strategies with the aim of finding an optimal performance. The remainder of this paper is structured as follows. After this introductory section, Section 2 reviews related work. Following, the proposed classification model is described in Section 3. Then, Section 4 presents the experimental results, and finally, Section 5 concludes the paper with a final discussion.

2 Related Work

Most of our literature review from the field is referenced by previous survey research (Fortuna and Nunes, 2018; Schmidt and Wiegand, 2017).

Multiple procedures have been implemented, since more traditional feature engineering, such us n-grams (Waseem and Hovy, 2016) or Part-of-Speech (POS) (Davidson et al., 2017) to more complex deep learning architectures (Yuan et al., 2016; Badjatiya et al., 2017).

According to the analyzed bias which motivates hate speech, general hate speech (Silva et al., 2016) is considered by the majority, however,

[1] https://ucr.fbi.gov/hate-crime/

there is large research that focuses particularly on racism (Kwok and Wang, 2013) and sexism (Hewitt et al., 2016). Though it is not exactly a form of hate speech, cyberbullying is a very related problem with some study research (Cortis and Handschuh, 2015).

3 System Overview

The system relies on a supervised machine learning algorithm. This final classification step is fed by a data processing pipeline formed by the preprocessing and the feature extraction modules. Regarding the implementation, Python has been used, with the additional capabilities provided by the libraries scikit-learn (Pedregosa et al., 2011), NLTK (Bird and Loper, 2004), and GSITK (Araque et al., 2017) [2]. Figure 1 illustrates the system architecture from a general perspective.

3.1 Preprocessing

In this phase, the raw text is taken and cleaned using common NLP techniques (Manning et al., 1999): removal of punctuation marks, special characters, URLs, and stop-words. Tweet preprocessing relies on tokenization, user mentions normalization, the appearance of hashtags, URLs, and all caps words flagged supported by the tools provided by GSITK. In addition, tokens are lemmatized using the Porter stemmer (Porter, 1980).

3.2 Feature Engineering

Different features have been taken into account during the feature engineering stage. Such features are divided into subcategories: statistical features, content analysis, word embeddings, semantic features, and linguistic features.

3.2.1 Statistical Features

We collected word and character n-grams evaluating both approaches, Bag-of-Words (BOW) and Term Frequency - Inverse Document Frequency (TF-IDF). The reason to include character n-grams comes from the Twitter domain, where texts are short and misspelling may occur; this can be attenuated at the character level (Schmidt and Wiegand, 2017). Apart from the mentioned reasoning, previous research (Mehdad and Tetreault, 2016) has shown the effectiveness of character n-grams in the problem of offensive language.

[2] https://github.com/gsi-upm/semeval2019-hatespeech

Besides tokens included within the text corpus, the system also includes frequencies from external lexicons that are thought for hate speech[3], sentiment analysis (Hu and Liu, 2004; Liu et al., 2005), and subjectivity analysis (Pang and Lee, 2004).

3.2.2 Content Analysis

As seen, sentiment and subjectivity information has been included. Hate speech can be considered as subjective content, and a relation between subjectivity, sentiments, and emotions can occur. Besides, hate speech is expected to have a negative polarity, so text subjectivity and polarity provided by the TextBlob (Loria et al., 2014) library were included in the analysis.

Topic modeling methods were added to the study, particularly, Latent Dirichlet Allocation (LDA) (Blei et al., 2003) in order to extract the topic of each tweet in combination with the appearance of hashtags (topics) inside the corpus.

3.2.3 Word Embeddings

In order to solve the lack of semantic of words in n-grams features, word distributed representations based on word embeddings models are evaluated. Pre-trained word vectors convert words into vector space where semantically similar words tend to appear close by each other. In this system, a vector is extracted for each word in the input text; then, as done in (Araque et al., 2017), the average pooling operation is performed on all word vectors, resulting in a vector of the same dimensions as the original word vectors.

3.2.4 Semantic Features

A central part of the system consists of a method (Araque et al., 2019) that exploits the semantic similarity measure that a word embedding model provides, via cosine similarity. In general lines, this approach uses a lexicon to which the input text is projected, employing the similarity measure obtained from an embedding model.

The method considers a selection of words S that constitutes a lexicon vocabulary to which the input documents are projected. Given a text document (e.g., tweet), a similarity value between the input word vectors of that document and each of the words in S is computed. After iterating over all input words and all lexicon words, a matrix $m \times |S|$ is obtained, where m is the number of input words in a particular document. Following,

[3] https://hatebase.org/

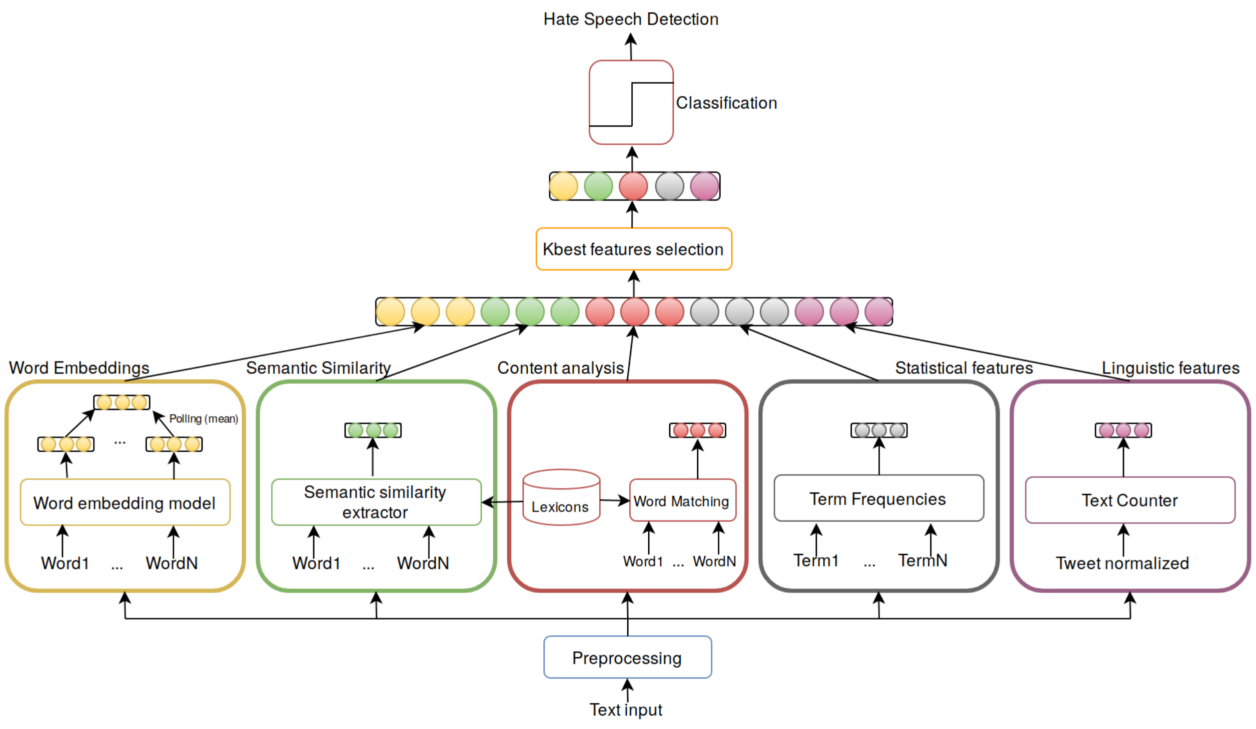


Figure 1: System Overview

the maximum pooling function is applied column-wise, obtaining the semantic similarity feature vector of dimensionality $|S|$.

In this work, the previously mentioned lexicons have been used, as well as a domain-oriented word selection, which have been extracted from the dataset. In this last approach, words were filtered by its frequency of appearance considering the document annotation, being the cutoff frequency an adjustable parameter.

3.2.5 Linguistic Features

The last set of features used are related to linguistic aspects. The proposed system considers the number of sentences, length from the tweet, POS stats, as well as some Twitter-related features such as the count of hashtags, URLs, mentions, all caps words, emojis, and exclamations.

3.3 Classification

Finally, the furthest step in the data processing pipeline makes use of a machine learning classifier. There are many options among machine learning models that can be used. In this project, we have evaluated the performance of three different types of algorithms: Logistic Regression, Support Vector Machines (SVM) with linear kernel, and Random Forest.

4 Experiments

This section presents the results obtained by the proposed system in the competition, considering both test and development phase submissions. Firstly, a data exploration has been carried out in order to analyze the data distribution, possible features to feed the classifier, and deficiencies in the data source. The evaluation of the different feature extraction approaches and the hyper-parameter tuning has been done by using a cross-validation grid search. Special attention has been paid in the regularization parameter of the algorithms: “C” parameter in the Logistic Regression and Linear SVM case and “maximum depth” of the trees in the Random Forest case. Finally, the system is trained, and the evaluation metrics are computed. This workflow has been repeated several times from the feature extraction step, changing the set of features in every iteration.

4.1 Sub-task A

The goal of this task is to classify both Spanish and English tweets as hateful or not hateful. Systems are evaluated using standard evaluation metrics, including accuracy, precision, recall, and F1-score, but predictions are ranked by F1-score metric alone.

Team	**Accuracy**	**Precision**	**Recall**	**F-score**
		English		
Best	0.506	0.65	0.566	0.457
SVM baseline	0.492	0.595	0.549	0.451
GSI-UPM	0.483	0.643	0.549	0.42
MFC baseline	0.579	0.289	0.5	0.367
		Spanish		
Best	0.731	0.734	0.741	0.73
GSI-UPM	0.728	0.726	0.733	0.725
SVM baseline	0.705	0.701	0.707	0.701
MFC baseline	0.58	0.294	0.5	0.37

Table 1: Official test set results for Task A

Feature combination	**Accuracy**	**Precision**	**Recall**	**F-score**
		English		
Official submission combination	0.777	0.774	0.780	0.775
Lexical, similarity, embeddings, and n-grams (1)	0.757	0.754	0.758	0.754
Bigrams, trigrams, similarity, and embeddings (2)	0.75	0.747	0.752	0.748
Embeddings, similarity, twitter stats, and LDA (3)	0.736	0.731	0.733	0.732
		Spanish		
Official submission combination	0.856	0.856	0.852	0.853
Lexical, similarity, embeddings, and n-grams (1)	0.812	0.811	0.807	0.808
Bigrams, trigrams, similarity, and embeddings (2)	0.796	0.794	0.791	0.792
Embeddings, similarity, Twitter stats, and LDA (3)	0.784	0.781	0.782	0.782

Table 2: Development set results for Task A

Task A data was partitioned into train, development, and test sets. Train and development sets were used to obtain the best feature combination by training over the train set and testing over the development one. Finally, for the final submission, the predictions for the test set were obtained with a system trained over both train and development sets.

Test results, which represent the official submission, as well as development phase results are presented in Tables 1 and 2 respectively. Task organizers included two baselines (Basile et al., 2019) in the competition, a linear SVM based on a TF-IDF representation and a trivial model that assigns the most frequent label from the training set to all instances in the test set.

The Spanish-oriented system relies on linguistic features (excepting POS), semantic similarity with a domain-oriented lexicon, sentiments (using the sentiment vocabulary weighted by the TF-IDF measure), word embeddings, topic modeling (both LDA and hashtags), and word and character TF-IDF n-grams. These features are filtered according to the ANOVA F-test, selecting the best 3,000. Linear SVM has been the selected machine learning algorithm for classification. On the other hand, the English-oriented system considers the same feature set excluding word embedding representation; the number of selected features has been set at 17,500. In contrast to the previous system, a Logistic Regression model was used to perform the classification.

Team	F-score(HS)	F-score(TR)	F-score(AG)	F-score (Avg)	EMR
English					
MFC baseline (Best)	0.367	0.452	0.445	0.421	0.58
GSI-UPM	0.421	0.686	0.556	0.555	0.312
SVM baseline	0.45	0.697	0.587	0.578	0.308
Spanish					
Best	0.729	0.798	0.737	0.755	0.705
GSI-UPM	0.725	0.79	0.735	0.75	0.624
SVM baseline	0.701	0.781	0.726	0.736	0.605
MFC baseline	0.37	0.424	0.413	0.402	0.588

Table 3: Official Results for Task B

Feature Combination	F-score(HS)	F-score(TR)	F-score(AG)	F-score (Avg)	EMR
English					
Official submission	0.775	0.811	0.723	0.770	0.665
(1)	0.754	0.797	0.712	0.755	0.641
(2)	0.748	0.788	0.699	0.745	0.628
(3)	0.731	0.767	0.687	0.728	0.611
Spanish					
Official submission	0.853	0.876	0.824	0.851	0.78
(1)	0.808	0.839	0.777	0.808	0.732
(2)	0.792	0.843	0.776	0.804	0.718
(3)	0.782	0.836	0.783	0.800	0.714

Table 4: Development Results for Task B

4.2 Sub-task B

The goal of this task is firstly to classify hateful tweets (i.e., tweets identified as hate speech against women or immigrants) as aggressive or not aggressive, and secondly to identify the target harassed as individual or generic (i.e., single person or group). Systems are evaluated by two criteria: partial match and exact match (Basile et al., 2019), but predictions are ranked by exact match metric alone.

For this task, the data has been delivered in the same way than sub-task A, so we emulated the same workflow than before, but in this case, considering solely hateful tweets. In this case, there are different distributions (Basile et al., 2019) along languages and sets, but different labels show a similar layout. This result goes in line with the work presented in (ElSherief et al., 2018), which states that directed hate speech is more informal, angrier, and often explicitly attacks the victim. Regarding the language, Spanish-speaking people tend to be more aggressive and more direct towards specific individuals. Seeing this skewed distribution, we outlined the idea to balance aggressiveness and directed messages by oversampling hateful tweets with not hateful ones, assuming that not hateful tweets are not aggressive nor directed.

As done previously, Tables 3 and 4 present official and development results, respectively. The Spanish-oriented system in this task is identical to that from Task A, but finally selecting 2,500 features. For the English case, in light of aggressiveness and target tweets, a different combination of features have been chosen. In order to detect aggressive tweets, all features except semantic similarity have been used, filtering the 32,500 best. Otherwise, for target messages, the complete set of features (sentiments and subjectivity were included by means of TF-IDF and semantic similarity) are used just considering the 2,500 best. Finally, different models were applied for each label, Logistic Regression for Target label and Linear SVM for the Aggressive one. The same algorithm selection was made in the Spanish case.

4.3 Discussion

In general terms, the results obtained are auspicious: the best submitted system achieved the 5th position in the Spanish Task A, 0.5% points under the best result obtained in the same task. For the Spanish Task B, the proposed model outperforms the baseline. In contrast to this, results in English Task A are lower than expected, where there was not any team that surpassed the 50% threshold in terms of F-score. As a general trend, test set results are worse than development results, which may indicate that our systems suffer over-fitting, and cannot generalize properly. This observation is enforced by attending to the English Task B, where no system has surpassed the baseline.

Since the data distribution is equal along languages in Task A (Basile et al., 2019), the difference in performance across languages may be due to Spanish speaking people are more explicit when typing any utterance with hate speech goals. As previously mentioned, we have observed that this type of hate speech messages show more aggressiveness. Language characteristics may be involved since the Spanish language has a morphologically-richer nature than English.

The presented results constitute the outcome of exhaustive experimentation of a variety of feature combination tests. In contrast with earlier work, semantic similarity and word embeddings representations do not produce such high performance results when compared to other domains such as sentiment analysis (Araque et al., 2019) and sleep disorder detection (Suarez et al., 2018) tasks. This circumstance suggests that hate speech detection is still an open challenge and more research must be done into the specific characteristics of such an exciting task.

Attending to the Spanish case, sentiment information and character n-grams were features that helped in a meaningful manner, confirming the issues raised in Sect. 3. For the English case, the improvement of the proposed features was incremental. While subjectivity and emojis had a relevant role in the results, this improvement was not as high as in the Spanish case. In light of the complexity of the hate speech domain, it could be argued that attending to word context instead of isolated words could help in the analysis. Indeed, n-grams include this type of information to some extent, but capturing the grammatical dependencies within a sentence (Chen, 2011) or template based strategies (Warner and Hirschberg, 2012) could enhance the performance.

5 Conclusions

This paper described the GSI-UPM hate speech detection system presented to participate in SemEval-2019 Task 5, which revolves around analyzing text messages from Twitter. In order to tackle this, a machine learning based approach has been developed. The different features that feed this system have been thoroughly evaluated, considering its suitability in the field of hate speech detection. It has been seen that both novel and traditional approaches do not yield so promising when used separately. Nevertheless, properly combining several types of features, as well as with content analysis features (e.g., sentiments and subjectivity) can improve the system to the point of reaching a reasonably good performance.

Concerning the achieved goals, the highest ranking was 5th place on the Spanish sub-task A, being 0.5% apart from the best performing system. This is, undoubtedly, a promising result that highlights the capacity of the proposed method to obtain nearly state-of-the-art performance in this task. When comparing with the same sub-task in the English case, in which we scored lower, it is necessary to study further the applicability of the system to different languages.

As future work, several lines of work could be addressed. Firstly, we plan to implement deep learning architectures which have shown to obtain better results in previous research (Zhang and Luo, 2018; Zhang et al., 2018). In addition, in order to afford imbalanced distributions, data augmentation (Hemker, 2018) techniques could be explored. Also, context-aware approaches could represent an improvement (Dinakar et al., 2012), since having general knowledge of hate speech (e.g., anti-LGBT or racism) may boost the performance of learning systems.

Acknowledgments

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness under the R&D project SEMOLA (TEC2015-68284-R) and the European Union with Trivalent (H2020 Action Grant No. 740934, SEC-06-FCT-2016).

References

Oscar Araque, Ignacio Corcuera-Platas, J. Fernando Snchez-Rada, and Carlos A. Iglesias. 2017. Enhancing deep learning sentiment analysis with ensemble techniques in social applications. *Expert Systems with Applications*, 77:236 – 246.

Oscar Araque, Ganggao Zhu, and Carlos A. Iglesias. 2019. A semantic similarity-based perspective of affect lexicons for sentiment analysis. *Knowledge-Based Systems*, 165:346 – 359.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, pages 759–760, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Steven Bird and Edward Loper. 2004. Nltk: The natural language toolkit. In *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*, ACLdemo '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Ying Chen. 2011. Detecting offensive language in social medias for protection of adolescent online safety.

Keith Cortis and Siegfried Handschuh. 2015. Analysis of cyberbullying tweets in trending world events. In *Proceedings of the 15th International Conference on Knowledge Technologies and Data-driven Business*, i-KNOW '15, pages 7:1–7:8, New York, NY, USA. ACM.

Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. *CoRR*, abs/1703.04009.

Karthik Dinakar, Birago Jones, Catherine Havasi, Henry Lieberman, and Rosalind Picard. 2012. Common sense reasoning for detection, prevention, and mitigation of cyberbullying. *ACM Trans. Interact. Intell. Syst.*, 2(3):18:1–18:30.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth M. Belding. 2018. Hate lingo: A target-based linguistic analysis of hate speech in social media. *CoRR*, abs/1804.04257.

Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Konstantin Hemker. 2018. Data augmentation and deep learning for hate speech detection. Master's thesis, Imperial College London.

Michael Herz and Peter Molnar. 2012. *The content and context of hate speech*. Cambridge University Press.

Sarah Hewitt, T. Tiropanis, and C. Bokhove. 2016. The problem of identifying misogynist language on twitter (and other online social spaces). In *Proceedings of the 8th ACM Conference on Web Science*, WebSci '16, pages 333–335, New York, NY, USA. ACM.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

Irene Kwok and Yuzhou Wang. 2013. Locate the hate: Detecting tweets against blacks. In *AAAI*.

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion observer: Analyzing and comparing opinions on the web. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 342–351, New York, NY, USA. ACM.

Steven Loria, P Keen, M Honnibal, R Yankovsky, D Karesh, E Dempsey, et al. 2014. Textblob: simplified text processing. *Secondary TextBlob: Simplified Text Processing*.

C.D. Manning, C.D. Manning, H. Schütze, and H.A. SCHUTZE. 1999. *Foundations of Statistical Natural Language Processing*. Mit Press. MIT Press.

Yashar Mehdad and Joel Tetreault. 2016. Do characters abuse more than words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

M.F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Leandro Araújo Silva, Mainack Mondal, Denzil Correa, Fabrício Benevenuto, and Ingmar Weber. 2016. Analyzing the targets of hate in online social media. In *ICWSM*, pages 687–690.

D. Suarez, O. Araque, and C. A. Iglesias. 2018. How well do spaniards sleep? analysis of sleep disorders based on twitter mining. In *2018 Fifth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 11–18.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, LSM '12, pages 19–26, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Shuhan Yuan, Xintao Wu, and Yang Xiang. 2016. A two phase deep learning model for identifying discrimination from tweets. In *EDBT*, pages 696–697.

Ziqi Zhang and Lei Luo. 2018. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *CoRR*, abs/1803.03662.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting hate speech on twitter using a convolution-gru based deep neural network. In *The Semantic Web*, pages 745–760, Cham. Springer International Publishing.

HATEMINER at SemEval-2019 Task 5: Hate speech detection against Immigrants and Women in Twitter using a Multinomial Naive Bayes Classifier

Nikhil Chakravartula
Teradata, Hyderabad
nikhil.chakravartula@gmail.com

Abstract

This paper describes our participation in the SemEval 2019 Task 5 - Multilingual Detection of Hate. This task aims to identify hate speech against two specific targets, immigrants and women. We compare and contrast the performance of different word and sentence level embeddings on the state-of-the-art classification algorithms. Our final submission is a Multinomial binarized Naive Bayes model for both the subtasks in the English version.

1 Introduction

Twitter is a micro-blogging platform where people exchange ideas using short messages called tweets. Users can propagate their notions, including hatred against an individual or a group, to the entire global population with a latency of a few seconds. This poses a unique challenge of developing systems that can automatically identify and mitigate hate speech. Although twitter condemns hate speech through its hateful conduct policy[1], enforcing it is difficult. There are several reasons for this. Tweets often contain emoticons, emojis, language slangs, hashtags and other noisy data. Often, offensive and abusive language may be erroneously perceived as hate speech and hence it is important to distinguish offensive, abusive and hateful languages (Davidson et al., 2017; Waseem and Hovy, 2016). These problems are exacerbated by the fact that even humans find it difficult to delineate offensive and hateful language.

Many approaches have been put forward to detect hate speech. Bag of words and ngram features are effective in hate speech detection (Burnap and Williams, 2015; Warner and Hirschberg, 2012) as well as the detection of abusive and offensive content (Nobata et al., 2016). Gitari et al. (2015) used lexical resources to look up certain words that contribute significantly to hate speech but such features, when used in isolation may not be very effective. SVM (Burnap and Williams, 2015), Naive Bayes (Kwok and Wang, 2013) and Logistic Regression (Davidson et al., 2017) are some of the classifiers used in this domain.

Most of the above methods are targeted to detect general hate speech. Through this task, we aim to identify hate speech, specifically against *immigrants and women*. Frenda et al. (2018) used lexicon resources to identify misogynistic comments. Ahluwalia et al. (2018) used an ensemble of random forest, gradient boosting and logistic regression with bag of words, ngram and lexical features to discern hatred against women. We did not find significant work in detection of hate speech in English against immigrants.

2 Shared Task Description

The SemEval 2019 Task 5 is divided into two subtasks.

1. Subtask A, where systems must predict whether a tweet is hateful (HS=1) against immigrants and women.
2. Subtask B, where systems must first classify hateful tweets as aggressive (AG=1) or not, and secondly to identify the target harassed as an individual (TR=1) or generic.

We used the datasets provided by the organizers. Table 1 describes the composition of the dataset. Further details of the task are available in the task description paper (Basile et al., 2019).

3 System Description

3.1 Pre Processing

We perform the following pre-processing operations on the text before feature engineering.

[1]https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy

	HS=1	TR=1	AG=1
Train	57.9%	17.3%	14.9%
Dev	42.7%	20.4%	21.9%

Table 1: Dataset composition. HS: Hate Speech TR: Target AG:Aggressiveness

- All text is converted to lower case.
- All URLs, mentions, emojis and smileys are removed from the tweets. We used a python package tweet-preprocessor[2] to achieve this.
- All contractions are replaced with their full form. For example, *don't* will be replaced by *do not* and *can't* will be replaced by *can not*.
- All punctuation marks are removed.
- All numerical sequences are removed from the text.
- **Hashtag segmentation and spell correction:** Hashtags provide insights about a specific ideology by a group of people. These notions provide vital information for text classification, especially in the case of hate speech against immigrants and women. For example, hashtags like *#endimmigration*, often come from a group of people who are against immigrants. Segmentation (Segaran and Hammerbacher, 2009) of the hashtags is essential to allow the classifier to treat *#buildthatwall*, *#buildthewall*, *#buildthedamnwall*, *#buildwall*, etc with the same importance. After segmentation, *#buildthatwall* becomes *build that wall*, *#buildthedamnwall* becomes *build the damn wall* etc. Many tweets contain abusive words in elongated form, such as *f****kkkk*. We perform spell corrections (Jurafsky and Martin, 2018) on these words to reduce the vocabulary size and to account for better results. Text8[3] is utilized to generate unigram and bigram word statistics with ekphrasis (Baziotis et al., 2017) to perform both these operations.
- **Stemming**: Stemming is the process of reducing a word to its base root form. We used Porter Stemmer[4] from NLTK (Steven Bird and Loper, 2009) to stem. Stemming is used in combination with the Naive Bayes classifier. For other classifiers, pretrained word embeddings without stemming are used.

[2]https://github.com/s/preprocessor

[3]http://mattmahoney.net/dc/textdata.html

[4]https://tartarus.org/martin/PorterStemmer/

3.2 Feature Engineering

The following features are considered in our experiments.

- **Bag of words (BoW):** Bag of words is used to represent the presence of word n-grams.
- **Word Embeddings:** Glove840B - common crawl, GloveTwitter27B - twitter crawl (Pennington et al., 2014) and fasttext - common crawl (Mikolov et al., 2018) pre-trained word embeddings are used to analyze their impact on the classification.
- **Sentence Embeddings:** Infersent (Conneau et al., 2017) is used to produce sentence level embeddings. InferSent is a sentence embedding method that provides semantic representations for English sentences. It is trained on natural language inference.

4 Experiments

In this section, we describe the experimental settings used in our research. All our code is publicly available in a github repository.[5]

4.1 Evaluation Metrics

The evaluation metrics for subtask A are precision(HS), recall(HS) and F_1-score(HS). Macro averaged F_1-score(HS,TR,AG) and Exact Match Ratio (EMR) are the evaluation metrics for subtask-B. Submissions are ranked based on F_1-score(HS) and EMR for subtask-A and subtask-B, respectively.

4.2 Methodology

All the experiments are developed using the Scikit-Learn (Pedregosa et al., 2011) machine learning library. Five-fold cross validation score on the train set used to evaluate our models. We ran several experiments on various classification algorithms. The best performing classifiers were Naive Bayes, logistic regression, SVM and XG-Boost. The following are the details of the classifier settings.

[5]https://git.io/fhFGR

WordVector	Logreg			SVM			XGB		
	F_{avg}(HS)	$F_{avg(HS,TR,AG)}$	EMR	F_{avg}(HS)	$F_{avg(HS,TR,AG)}$	EMR	F_{avg}(HS)	$F_{avg(HS,TR,AG)}$	EMR
glove	0.58	0.57	0.49	0.54	0.54	0.45	0.61	0.56	0.44
fasttext	0.58	0.56	**0.53**	0.55	0.52	0.45	0.61	0.56	0.43
glove twitter	0.69	0.67	0.48	**0.69**	0.61	0.45	0.69	0.65	0.44
glove + infersent	**0.73**	**0.70**	0.47	0.66	**0.64**	**0.46**	**0.72**	**0.68**	**0.46**

Table 2: Pretrained word and sentence embeddings results. For each classifier family, the best score is made bold.

Word ngrams	Stem	Binary	$F_{avg(HS)}$	F_{avg} (HS,TR,AG)	EMR
1,2	false	true	0.69	0.66	0.45
1,2	false	false	0.68	0.66	0.45
1,2	true	true	**0.69**	**0.67**	**0.47**
1,2	true	false	0.68	0.66	0.45
1,3	false	true	0.69	0.66	0.45
1,3	true	true	0.69	0.66	0.47
1,4	false	true	0.69	0.66	0.45
1,4	true	true	0.69	0.66	0.47

Table 3: Multinomial Naive Bayes Classifier results with word ngram range, stemming and binarization

Classifier	$F_{1(HS)}$	F_1 (HS,TR,AG)	emr
glove twitter+logreg	0.70	0.70	0.48
glove twitter+XGB	0.69	0.66	0.55
glove+infersent+ XGB	0.72	0.69	0.55
glove+infersent+ logreg	0.72	0.72	0.53
stem+ NB binarized+ word-ngrams(1.2)	**0.74**	**0.73**	**0.57**

Table 4: Results on the dev set

- **Logistic Regression, SVM and XGBoost** Word or sentence level embeddings are fed as inputs to these classifiers. In the absence of a sentence embedder, we averaged all the word vectors to get a vector representation of the tweet. For logistic regression, the solver is liblinear (Fan et al., 2008) and L2 norm is used for penalization. For SVM, inputs are normalized using a soft scaling scheme and the kernel used is a Radial Basis Function (Buhmann and Buhmann, 2003). The default parameters are kept as is for XGBoost[6]. Table 2 shows the results.

- **Naive Bayes classifier:** Multinomial Naive Bayes classifier, along with the bag of words generated with CountVectorizer[7] gave better results than other Naive Bayes variations. Different runs are carried out to tune the parameters as shown in Table 3.

[6]https://xgboost.readthedocs.io/en/latest/python/python_api.html

[7]https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

5 Results and Analysis

We wanted to submit a single system for both the subtasks. Hence, our goal was to maximize all three metrics: F_1(HS), F_1(HS,TR,AG) and EMR. The results show that there is no single variation that defeats the others in all the metrics combined. Logistic regression with glove and infersent performed the best in F_1(HS) and F_1(HS,TR,AG), but only with an acceptable EMR. Regarding the XGBoost family, glove with inferesent version outperforms the rest in all the metrics. Stemmed-binarized Naive Bayes classifier with ngram range (1,2) performed better in F_1(HS) and EMR in the Naive Bayes family. The Glove-Twitter version of logreg and XGboost aren't too far behind as well. We applied all these high performing models on the dev set to analyse their performance further. The results are shown in Table 4. Naive Bayes comfortably achieved the highest score on the dev set on all the three metrics as shown in Table 4. Hence, we finalized the Naive Bayes model as our official submission. This submission scored an F_1(HS) of 0.405 in subtask-A, F_1(HS,TR,AG) of 0.54 and EMR of 0.296 in subtask-B.

6 Conclusion and Future Work

The aim of this research was to detect hate speech against two specific targets, immigrants and women. We described a naive bayes classifier system and also elucidated our trials of using different pre-trained word and sentence level embeddings on the state-of-the-art classification algorithms. In the future, we would like to include lexicon-based, Parts Of Speech features to further investigate the performance of these classifiers. We would also like to evaluate how deep learning approaches respond to this task.

References

Resham Ahluwalia, Himani Soni, Edward Callow, A. Nascimento, and Martine De Cock. 2018. Detecting hate speech against women in english tweets. In *EVALITA@CLiC-it*.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics", location = "Minneapolis, Minnesota.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada. Association for Computational Linguistics.

Martin D. Buhmann and M. D. Buhmann. 2003. *Radial Basis Functions*. Cambridge University Press, New York, NY, USA.

Pete Burnap and Matthew L. Williams. 2015. Cyber hate speech on twitter : An application of machine classification and statistical modeling for policy and decision making.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. *CoRR*, abs/1703.04009.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Simona Frenda, Bilal Ghanem, Estefanía Guzmán-Falcón, Manuel Montes-y-Gómez, and Luis Villaseñor Pineda. 2018. Automatic expansion of lexicons for multilingual misogyny detection. In *Proceedings of the Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018) co-located with the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018), Turin, Italy, December 12-13, 2018.*

Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long. 2015. A lexicon-based approach for hate speech detection.

Daniel Jurafsky and James H. Martin. 2018. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall.

Irene Kwok and Yuzhou Wang. 2013. Locate the hate: detecting tweets against blacks. In *AAAI 2013*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 145–153, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Toby Segaran and Jeff Hammerbacher. 2009. *Beautiful Data: The Stories Behind Elegant Data Solutions*. O'Reilly Media, Inc.

Ewan Klein Steven Bird and Edward Loper. 2009. *Natural Language Processing with Python– Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, Inc.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, LSM '12, pages 19–26, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93. Association for Computational Linguistics.

HATERecognizer at SemEval-2019 Task 5: Using Features and Neural Networks to Face Hate Recognition

Victor Nina-Alcocer
Department of Computer Systems and Computation
Universitat Politècnica de València, Spain
vicnial@inf.upv.es

Abstract

This paper presents a detailed description of our participation in task 5 on SemEval-2019[1]. This task consists of classifying English and Spanish tweets that contain hate towards women or immigrants. We carried out several experiments; for a finer-grained study of the task, we analyzed different features and designing architectures of neural networks. Additionally, to face the lack of hate content in tweets, we include data augmentation as a technique to increase hate content in our datasets.

1. Introduction

HatEval (Basile et al., 2019) aims to identify cases of aggressiveness and hate speech towards women and immigrants in social media considering tweets messages written in English and Spanish. This task defines two main sub-tasks:

- Task1. Hate Speech Detection against Immigrants and Women: predict whether a tweet in English or in Spanish is hateful or not hateful.
- Task2. Aggressive behavior and Target Classification: to identify if a tweet is aggressive or not aggressive, and to identify the target harassed as individual or generic.

To tackle the subtasks mentioned above we will use Natural Language Processing (NLP) and Machine Learning (ML) fields to propose two approaches. The first approach pretends to know more about valuable features that allow us to get a good understanding of its embedded knowledge. To get this knowledge we study four features that we consider important: its structure, its embedded emotions, patterns on its pos tagging and skip-grams. Each of these features will allow us to know if tweets have some patterns that can be used to discriminate tweets that contain hate or not.

The second approach will try to recognize patterns using weights among neurons. To implement this focus we designed several architectures of neural networks (NN), which were fed with different kinds of corpora that were processed considering many aspects such as, lemmatization, stemming, normalized hashtags, etc.

This work is organized into three sections. The first section provides an introduction to this paper. The second one describes the proposed approaches, the experiments conducted, and the results that we achieved. The third section presents some conclusions.

2. Systems Description

In this section, we present the main features of the two approaches considered in this paper and its respective experiments.

The organizers provided a training dataset of 9000 and 4500 tweets written in English and Spanish labeled with hate speech (HS), Aggressive behavior (AG) and Target (TR). For what concerns HS the distribution is almost balanced among hate(42 %) and no-hate (58 %) tweets in both languages. Regarding AG and TR, the distribution is skewed towards tweets that do not contain TR (75 %) or AG(67 %) respectively. In order to asses, the performance of the systems, tests set of 3000 and 1600 unlabeled tweets were provided. The official evaluation metrics to evaluate the systems were: For the task1, accuracy (Acc.), precision (P), recall (R), and F1-score. For task2, The models were evaluated using EMR and F1-score as describe in (Basile et al., 2019)

2.1. Based on Classical ML

With this approach, we try to face hate detection through classical machine learning algo-

[1]http://alt.qcri.org/semeval2019/index.php?id=tasks

rithms. Firstly, we established a baseline (called «our baseline») using Term frequency - Inverse document frequency (TF-IDF) scheme based on words and Support Vector Machine (SVM) as a supervised learning model. This baseline gives us an idea of whether the experiments that consider new features or simply the use of other machine learning algorithms help us to achieve good results.

Preprocessing: The preprocessing consisted of ridding of URLs, numbers, users, times, date, email, percents. But, we decided to keep normalized hashtags, elongated words, repetitions, emphasis, and censored words. Next, we present an example to show a raw and normalized version of a tweet:

Raw tweet:

@KamalaHarris Illegalsssssssss Dump their Kids at the border like Road Kill and Refuse to Unite! They Hope they get Amnesty, Free Educatioon and Welfare Illegal #FamilesBelongTogether in their Country not on the Taxpayer Dime Its a SCAM #NoDACA #NoAmnesty #SendThe https://t.co/Ks0SHbtYqn[2]

Normalized tweet:

illegals dump their kids at the border like road kill and refuse to unite they hope they get amnesty free education and welfare illegal familesbelongtogether in their country not on the taxpayer dime its a scam nodaca noamnesty sendthe

Feature Extraction and Selection. As in (Schmidt and Wiegand, 2017) our models considered a set of features. The first one takes into account the *structure of tweets*, with this focus we are going to consider and count if tweets have hashtags, punctuation marks, presence and its *Unicode Common Locale Data Repository* (CLDR)[3] version of emojis, capital letters, number of words, numbers of user and so on. To get the second group of features was necessary to consider *pos tagging* as described (Bretschneider et al., 2014), but analyzing some patterns on how people usually write or use hate content in tweets as was mentioned by (Silva et al., 2016), for instance, we noticed in the dataset that people usually use this pattern to denigrate a woman: "eres una maldita zorra" now to discover some pattern we show its pos tagging representation below:

eres ***AUX*** *una* ***DET*** *maldita* ***ADJ_Gender=Fem*** *puta* ***NOUN_Gender=Fem*** *zorra* ***ADJ_Gender=Fem*** *.* ***PUNCT_PunctType=Peri***

As we know a sentence is composed by verbs, prepositions, adjectives, adverbs and so on. In this particular sentence shown above, we see that is composed by auxiliaries, determiner, adjectives, noun and a punctuation mark. All these tags plus the gender of each word will give us a piece of important information to discriminate hate towards women and immigrants. The third feature that we consider important is to try to *find out if a word is pejorative, derogative, offensive or vulgar*. To get this aim we created a lexicon with these four categories already mentioned before. In short, we selected words that we called "seeds" then we start to get synonyms or related words using several sources[4]. Using this method we got more than 4900 words classified in the four mentioned categories. Some of the words and its respective categories are shown in Table 1.

Word	Pejorative	Derogative	Offensive	Vulgar
* solterona (es)	✓	-	-	-
* puta/puta de quinta (es)	✓	-	✓	-
* idiota (es)	-	✓	-	-
* huevon (es)	-	-	✓	-
* asswhore (en)	-	-	✓	-
* fucknigga/nigga (en)	-	-	✓	-
* pisslamist (en)	-	-	-	✓
* dogfucker (en)	-	-	-	✓

Table 1: Categories considered in our hate_lexicon.

We noticed that extending a little bit more the four categories in the third feature, we can improve slightly our results. Therefore, we take into account *the percentage of embedded emotions* in tweets, for this goal we used Linguistic Inquiry and Word Count (LIWC)[5]. And we chose some additional categories (sexual, anxiety feeling, anger and so on.) that help us to discriminate between tweets which contents hate or not. And finally, we have considered using TF-IDF schemes over skip-grams (bigram and trigrams) (Davidson et al., 2017).

In these experiments, we used scikit-learn to test several machine learning algorithms and its respective parameters. Some algorithms that we used

[2]This raw tweet belongs to the original training dataset and it was intentionally rewritten with typos to illustrate which considerations were taken into account to its normalization.

[3]http://cldr.unicode.org/translation/short_names_and_keywords

[4]https://www.dictionary.com

[5]https://www.receptiviti.ai/liwc-api-get-started

Approach	**Task1**	**Task1**	**Task2**	**Task2**	**Task1**	**Task1**	**Task1**	**Task1**
	English	**English**	**English**	**English**	**Spanish**	**Spanish**	**Spanish**	**Spanish**
	Acc.	**F1-sc.**	**F1-sc.(avg)**	**EMR**	**Acc.**	**F1-sc.**	**F1-sc.(avg)**	**EMR**
OUR BASELINE(See section 2.1)	*0.763*	*0.766*	*0.731*	*0.629*	*0.834*	*0.840*	*0.785*	*0.710*
TFIDF+POS	0.769	0.770	0.740	0.633	0.840	0.837	**0.792**	**0.719**
TFIDF+EMOTIONS	0.759	0.749	0.747	0.711	**0.85**	**0.847**	0.788	0.698
TFIDF+STRUCT	**0.771**	**0.780**	**0.75**	**0.648**	0.790	0.798	0.789	0.702
TFIDF+POS+EMOTIONS	↓	↓	↓	0	0.849	0.842	0.791	0.700
TFIDF+POS+STRUCT	↓	↓	↓	0	0.810	0.800	0.787	0.700
TFIDF+EMOTIONS+POS	↓	↓	↓	0	0.846	0.839	0.781	0.698
TFIDF+SKIP-BIGR.	↓	↓	↓	↓	↓	↓	↓	↓
TFIDF+SKIP-TRIGR.	↓	↓	↓	↓	↓	↓	↓	↓
TFIDF+SKIP-BIGR.+EMOTIONS	↓	↓	↓	↓	↓	↓	↓	↓
TFIDF+SKIP-BIGR.+STRUCT	↓	↓	↓	↓	↓	↓	↓	↓
TFIDF+SKIP-BIGR.+EMOTIONS+STRUCT	↓	↓	↓	↓	↓	↓	↓	↓
TFIDF+SKIP-TRIGR.+EMOTIONS	↓	↓	↓	↓	↓	↓	↓	↓
TFIDF+SKIP-TRIGR.+STRUCT	↓	↓	↓	↓	↓	↓	↓	↓
TFIDF+SKIP-TRIGR.+EMOTIONS+STRUCT	↓	↓	↓	↓	↓	↓	↓	↓

Table 2: Results achieved using LinearSVC algorithm for English and Spanish development datasets.

were: RandomForestClassifier (Breiman, 2001), LogisticRegression, Naive Bayes (Kibriya et al., 2004), LinearSVC (Hearst, 1998) and so on (Fortuna, 2017). A summary of applying this approach to the developing dataset and its respective results are shown in Table 2. This table shows just results using LinearSVC because it was the algorithm that achieved good results. Other variations of corpus like stemming or lemmatization version were not used because we did not achieve good results.

Basically, we can highlight in table 2, that the use of skip-grams does not help a lot, this fact is due to the lack of skip-grams contained in tweets, for instance, in the tweet mentioned before, the skip-trigram «illegals dump kids» is just repeated once in the whole dataset. A similar situation happens with skip-bigrams and skip-trigrams. On the other hand, the use of the structure and emotions (lexicon) embedded in tweets help to improve our baseline in both languages respectively. We used down arrow symbol ↓ to show that the value is below of our baseline, so is not valuable include it.

2.2. Based on NN

This second approach tackle hate speech detection trough neural networks (Schmidt and Wiegand, 2017). In this stage, we used several neural networks architectures which works enough well with text, some of them are: Convolutional Neural Networks (CNN) (Jacovi et al., 2018), Long short term memory (LSTM) and its variations: Peephole, Bidirectional (BILSTM) and Gated Recurrent Unit (GRU) (Lu and Salem, 2017). We tested these architectures with different corpora which were slightly modified, in some cases we used a corpus with stop words, or simply a stemming or a lemmatized version of words. Additionally, we used other variations to see which of them help us to improve the performance of the system, the next paragraph describes how the corpora were processed.

Preprocessing: Is valuable to mention that the preprocessing of raw tweets was slightly different for machine learning and NN, because we saw that keeping an identical preprocess for both approaches do not help a lot. So we decided to customize the preprocess for each focus as we describe in section 2.1 and section 2.2. Swear words were kept for both focuses because they helped to achieve better results. Furthermore, emojis were normalized using its CLDR short name, For instance, we changed 😊 for ':smiling_face_with_smiling_eyes:' cleaned (without any additional symbol).

For this particular preprocess for NN were taken to account many aspects that were already considered for ML's preprocessing. Perhaps the main difference is when we process the hashtag. For instance, its version normalized for the raw tweet shown in section 2.1, would be:

illegals dump their kids at the border like *road kill and refuse to unite they hope they get amnesty free education and welfare illegal families belong together in their country not on the taxpayer dime its a scam no daca no amnesty send the*

As is shown, this version separates the content of hashtags, additionally, we correct typos using dictionaries[6]. For instance, we changed "familes" by "families" in the above tweet.

This preprocess used for NN was used for ML too, but we noticed that accuracy got down in our experiments. Then we decided to keep the original way of hashtags for ML (no splitting).

The results shown in Table 3 were achieved using the developing datatset with the preprocess commented in section 2.2. Summing up, stop words, normal words (no stemming, no lemmatization), normalized hashtags, spelling corrections were considered. Furthermore, word embeddings (Mikolov et al., 2013) (Goldberg, 2015) as efficient word representation were used for all the experiments in this stage.

Carrying on several experiments we realized that Convolutional Neural Networks performed well enough compared to other architectures, see Table 3. Therefore, we named this architecture as *architecture A*. This architecture is compound by an input embedding layer, followed by and spatial dropout and next to a convolutional layer with a set up of 256 filters and kernel size of 2. Next a globalmaxpooling is defined and finally, a dense layer of 2 neurons with softmax is used to get the results. The good performance of this architecture encouraged us to use different word embeddings to improve accuracy, the results of these experiments are shown in Table 4.

As we can see in Table 4, there is a lack of results on the Spanish language, this fact is due that we could find in Spanish all the corresponding resources exploited for experiments with English portion of the data. Sometimes some authors publish word embeddings for this specific language but mostly this embeddings does not fit with our study case due to the language used in twitter is kind of informal and use a lot of slang, swear words and so on. And this important aspect reduces the performance of our system.

Data Augmentation: As we commented at the beginning of section 2, we noticed a slight skew in favor of tweets that do not contain HS, AG or TR. To deal with this unbalanced training dataset, like in (Risch and Krestel, 2018) we adopted a technique which allows us to increase the number of tweets with hate content. To do this we analyzed some ways to get synonyms or similar words. Firstly, we try to use synonyms only if these synonyms do not change the meaning of the whole sentence. This fact is challenging because is kind of hard to use synonyms to increase our dataset. For instance, the next tweet: *"Hurray, saving us $$$ in so many ways potus realDonaldTrump #LockThemUp #BuildTheWall #EndDACA #BoycottNFL #BoycottNike"* has to be cleaned and it has to just keep words to get its synonyms. The result to process the tweet above is: *"hurray saving us in so many ways lock them up build the wall end daca boycott nfl boycott nike"*. Now, we just have to get synonyms for each word, in our case using some resources such as: wordnet, dictionaries and so on. It is important to highlight that wordnet and dictionaries contain a formal language and those have a lack of slang or informal language used in tweets. Therefore, we decided to use word embeddings to get the most similar words, Additionally, we defined a threshold[7] to control that the synonym was so close to the original word. In our case, the threshold has been defined as *0.7* and the results for the tweet mentioned before are shown in Table 5.

Some remarkable facts that we saw in our experiments are: if we keep a threshold over 0.9 for sure we would not be able to get synonyms because with a high threshold we are just recovering almost the same words used on the original sentence. On the other hand, if we keep a threshold under 0.5 we will get synonyms or similar words that are not coherent and additionally those new words change the meaning of the whole sentence. For instance, in the syn3 with a threshold 0.3, the original word (us) was changed by (states), we assume that it refers to the country called the United States, that sometimes is written like *us*, this is just an example of how many of these words change the meaning of the sentence when is used a lower threshold. Unfortunately, this approach was not able to achieve good results. But we noticed that is valuable go deeper with this focus.

[6]https://www.dictionary.com

[7]most_similar function return the top n similar words and their respective scores. In our case we assume, if the score is close to 1 then more similar is the word.

Architecture	Spanish						English					
	Task1				Task2		Task1				Task2	
	P	R	A	F1-score	F1-score(avg)	EMR	P	R	A	F1-score	F1-score(avg)	EMR
Emb.+CNN1d	0.84	0.84	0.84	0.84	0.78	0.69	0.74	0.73	0.74	0.72	0.74	0.62
Emb.+CNN1d*4+dense*2	0.70	0.70	0.70	0.70	0.67	0.58	0.63	0.64	0.64	0.64	0.62	0.51
Emb.+LSTM	0.78	0.77	0.78	0.77	0.75	0.67	0.61	0.63	0.63	0.62	0.73	0.64
Emb.+LSTM_Peep	0.77	0.77	0.77	0.77	0.74	0.65	0.57	0.57	0.57	0.58	0.70	0.58
Emb.+BILSTM	0.79	0.79	0.79	0.79	0.75	0.66	0.70	0.70	0.71	0.70	0.71	0.59
Emb.+BILSTM_GRU	0.81	0.81	0.81	0.81	0.75	0.66	0.73	0.73	0.73	0.74	0.72	0.63

Table 3: Results achieved using several NN architectures for English and Spanish development datasets.

	Task1			
	Spanish		English	
	Acc.	F1-score	Acc.	F1-score
glove+100	-	-	0.72	0.719
glove+200	-	-	0.74	0.745
glove+300	0.842	0.84	0.739	0.73
google+300	-	-	0.733	0.728
fasttext+300	0.777	0.78	0.732	0.75

Table 4: Results achieved by applying word embeddings to architecture A over development dataset.

Threshold	Original Sentence
	hurray saving us in so many ways lock them up build the wall end daca boycott nfl boycott nike
Threshold	**Similar Sentence**
0.9	syn1: 'hurray saving us in so many ways lock them up build the wall end daca boycott nfl boycott nike'
0.9	syn2: 'hurray saving us in so many ways lock them up build the wall end daca boycott nfl boycott nike'
0.9	syn3: 'hurray saving us in so many ways lock them up build the wall end daca boycott nfl boycott nike'
0.7	syn1: 'hurray save us in but several ways locks themselves up construct of walls end daca boycotting redskins boycotting adidas'
0.7	syn2: 'hurray saved us in anyway numerous ways lock them up rebuild the wall end daca boycotts usfl boycotts nike'
0.7	syn3: 'hurray saving us in even countless ways lock them up build the wall end daca boycotted packers boycotted nike'
0.5	syn1: 'hurra save u the but several way locks themselves ups construct of walls beginning daca boycotting redskins boycotting adidas'
0.5	syn2: 'hurrah saved us/uk , anyway numerous things locking they up/ rebuild which ramparts ends daca boycotts usfl boycotts reebok'
0.5	syn3: 'hooray rescuing states where even countless possibilities padlocks those messed constructing and doorway ending daca boycotted packers boycotted sportswear'
0.3	syn1: 'hurra save u the but several way locks themselves ups construct of walls beginning să boycotting redskins boycotting adidas'
0.3	syn2: 'hurrah saved us/uk , anyway numerous things locking they up/ rebuild which ramparts ends nica boycotts usfl boycotts reebok'
0.3	syn3: 'hooray rescuing states where even countless possibilities padlocks those messed constructing and doorway ending bucuresti boycotted packers boycotted sportswear'

Table 5: Similar words that were gotten using word embeddings.

Official Ranking

Table 6 shows the official results published by the organizers of HatEval. We submitted 10 runs for each task, the runs that achieved good results were run01 and run07 for English and run05 and run10 for Spanish. As we see too the performance of our the four runs or systems achieved high values over their respective baselines. Run01 and Run05 use TF-IDF scheme, run01 uses just structure features, meanwhile, run05 uses pos tagging and embedded emotions. Features that we have already described in section 2.1. Both runs achieved good results using Support Vector Machine. The system used in run07 is the *architecture A* which was already described in section 2.2. Run10 used an embedding layer as input, followed by an LSTMPeephole layer and finally a dense layer of 2 neurons with softmax is used. The last two runs that use neural networks were fed with a corpus described in section 2.2. As we notice both approaches perform enough well, but architectures that use neural networks work slightly better than focuses that use classical machine learning algorithms.

	Task1		**Task2**	
	English			
	Acc.	**F1-score(avg)**	**EMR**	**F1-score(avg)**
SVC off. Base.	0.492	0.451	0.308	0.578
vmna (run01)	0.512	0.472	0.341	0.577
vmna (run07)	0.515	0.481	0.353	0.569
	Spanish			
SVC off. Base.	0.705	0.701	0.605	0.736
vmna (run05)	0.721	0.718	0.621	0.739
vmna (run10)	0.735	0.729	0.632	0.747

Table 6: Official results for task1 and task2.

Having the golden labels we realize that our proposed system misclassified several tweets. As we can see in the next examples:

1. ** (original tweet) Estas navidades mi polla mereces*
   ** (Engish version) you deserve my dick for Christmas*
   Language: Spanish
   Golden Label: no-hate (0)
   our system(run05): hate (1)

2. ** (original tweet) @shakhepri69 @KuriMelon21 @HoruSenpai @crafter657 oye puta basura, cállate*
   ** (Engish version) @shakhepri69 @KuriMelon21 @HoruSenpai @crafter657 shut the fuck up, bitch*
   Language: Spanish
   Golden Label: no-hate (0)
   our system(run05): hate (1)

3. ** (original tweet) @GMA @TVMarci His own fault #SENDTHEMBACK*
   ** (Spanish version) @GMA @TVMarci su culpa #SENDTHEMBACK*
   Language: English
   Golden Label: no-hate (0)
   our system(run01): hate (1)

All three examples have been labeled as hate by our classifier. On the first example, after preprocessing, the system just kept the words navidades (Christmas), polla (dick) and mereces (deserve). And looking for these words in our lexicons just the last two words were found, so the system marked the tweet as hate-content. The same situation happened with example 2 and 3, just the words puta (bitch), basura (trash), callate (shut up) and culpa (fault), send (enviar), back (regreso) were found. As (Zhang and Luo, 2018) noticed, that some times hate and no-hate tweets have no enough features to be differentiated. Doing a manual analysis in the datasets we notice that fact to in several tweets.

3. Conclusions

This work proposed two main approaches. The first one, take into account some features that can be considered to feed classical machine learning algorithms, some of these features are: structure, embedded emotions, pos tagging, and skip-grams. On the other hand, the second approach consists of designing several neural networks architectures to test a variety of corpora and word representations. Moreover, we face an unbalanced dataset using a technique called data augmentation. To get similar words were used pre-trained word embeddings with a tuned few thresholds. Using data augmentation our results did not improve but we noticed that is a promising field to study. Furthermore, looking at the results in Table 6 we appreciate that both approaches contribute to achieving good results, perhaps a deep study in both approaches can help us to understand and improve our results. As a future work, it is interesting to explore more deep learning or neural networks and design more complex architectures.

References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Se-*

mantic Evaluation (SemEval-2019). Association for Computational Linguistics.

Leo Breiman. 2001. Random forests. *Mach. Learn.*, 45(1):5–32.

Uwe Bretschneider, Thomas Wöhner, and Ralf Peters. 2014. Detecting online harassment in social networks. In *ICIS*.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language.

Paula Fortuna. 2017. FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO Automatic detection of hate speech in text: an overview of the topic and dataset annotation with hierarchical classes. Technical report.

Yoav Goldberg. 2015. A primer on neural network models for natural language processing. *CoRR*, abs/1510.00726.

Marti A. Hearst. 1998. Support vector machines. *IEEE Intelligent Systems*, 13(4):18–28.

Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg. 2018. Understanding convolutional neural networks for text classification. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 56–65. Association for Computational Linguistics.

Ashraf M. Kibriya, Eibe Frank, Bernhard Pfahringer, and Geoffrey Holmes. 2004. Multinomial naive bayes for text categorization revisited. In *Proceedings of the 17th Australian Joint Conference on Advances in Artificial Intelligence*, AI'04, pages 488–499, Berlin, Heidelberg. Springer-Verlag.

Yuzhen Lu and Fathi M. Salem. 2017. Simplified gating in long short-term memory (LSTM) recurrent neural networks. *CoRR*, abs/1701.03441.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Julian Risch and Ralf Krestel. 2018. Aggression identification using deep learning and data augmentation.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10. Association for Computational Linguistics.

Leandro Silva, Mainack Mondal, Denzil Correa, Fabricio Benevenuto, and Ingmar Weber. 2016. Analyzing the Targets of Hate in Online Social Media.

Ziqi Zhang and Lei Luo. 2018. Hate Speech Detection: A Solved Problem? The Challenging Case of Long Tail on Twitter.

GL at SemEval-2019 Task 5: Identifying hateful tweets with a deep learning approach.

Gretel Liz De la Peña Sarracén
Universitat Politècnica de València, Spain

Abstract

This paper describes the system we developed for SemEval 2019 on Multilingual detection of hate speech against immigrants and women in Twitter (HatEval - Task 5). We use an approach based on an Attention-based Long Short-Term Memory Recurrent Neural Network. In particular, we build a Bidirectional LSTM to extract information from the word embeddings over the sentence, then apply attention over the hidden states and finally feed this vector to another LSTM model to get a representation from de data. Then, the output obtained with this model is used to get the prediction of each of the sub-tasks with models based on neural networks and linguistic characteristics.

1 Introduction

Nowadays, the number of content generated by users on social networks is growing rapidly. In this context, the problem of detecting and limiting the dissemination of the Hate Speech is becoming a matter of great importance. Therefore, many efforts are dedicated to studying and treating this phenomenon. A large number of workshops on this topic have been developed in recent years, which reflects the interest of many researchers.

Some examples are the Workshop on Trolling, Aggression and Cyberbullying (Kumar et al., 2018), that included a shared task on aggression identification; the tracks on Automatic Misogyny Identification (AMI) (Fersini et al., 2018a) and on Autohorship and Aggressiveness Analysis (MEX-A3T) (Álvarez-Carmona et al., 2018) proposed at IberEval 2018; the Automatic Misogyny Identificationtask at EVALITA 2018 (Fersini et al., 2018b), the Workshop on Abusive Language (Waseem et al., 2017) and the GermEval Shared Task on the Identification of Offensive Language (Wiegand et al., 2018).

The proposed works have used different features and models. Among them, models based on deep learning, such as Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) have been widely used.

This paper presents a strategy based on RNN, which is an extension of previous models proposed for the tasks MEX-A3T and EVALITA 2018 (Cuza et al., 2018; la Peña Sarracén et al., 2018). Those models use an Attention-based LSTM inspired by the work (Yang et al., 2016). In this work the authors use a hierarchical attention network for document classification and their experiments show that the architecture outperforms previous methods. This last model has two levels of attention mechanisms applied at the word and sentence level, enabling it to attend differentially to more and less important content when constructing the document representation.

The aim of the proposed extension in this work is not only the hate language identification, but also the study of other features of hateful messages. In this case, the model based on the Attention mechanism and the LSTM is used as a representation of the input. This representation is then used to detect the hate language with a fully connected network. In addition, it is combined with some linguistic characteristics to analyze the other features of hateful messages. The objective has been defined for the HatEval[1] shared task on SemEval 2019.

The HatEval (Basile et al., 2019) task consists in Hate Speech identification in messages from Twitter in Spanish and English. The main objective focuses on detecting the hate expressed against women and immigrants in particular. The task is divided into two related subtasks: a binary classification as a first sub-task about Hate Speech de-

[1]https://competitions.codalab.org/competitions/19935

tection, and another one where other features of hateful contents is investigated:

- TASK A - Hate Speech Detection against Immigrants and Women: Predicting whether a tweet with a given target (women or immigrants) is hateful or not (HS).
- TASK B - Aggressive behavior and Target Classification: Classifying hateful tweets as aggressive or not (AG), and second identifying the target harassed as individual or generic (TR).

The paper is organized as follows. Section 2 describes the proposed methodology. Experimental results are then discussed in Section3. Finally, we present our conclusions with a summary of our findings in Section 4.

2 Methodology

In this work, a simple preprocessing is performed in which the text is cleaned. First, emoticons, hashtags, urls, and other strings that do not represent alphabetic sequences are eliminated. Then, the texts are represented as vectors with a word embedding model. We used pre-trained word vectors of Glove (Pennington et al., 2014), trained on 2 billion words from Twitter for English. On the other hand, for Spanish we used the word vectors of fasttext (Bojanowski et al., 2017).

In general, we propose a model that consists of a Bidirectional LSTM neural network (Bi-LSTM) at the word level in the input. At each time step the Bi-LSTM gets as input a word embedding. Afterward, an attention layer is applied over each hidden state. The attention weights are learned using the concatenation of the current hidden state of the Bi-LSTM and the past hidden state of another LSTM. In this way, a representation (R) of the input is obtained from the last LSTM to get the prediction of each of the subtasks, as shown in Figure 1.

2.1 Sub-task A

For the sub-task A, which consists in the identification of hate in tweets (HS), R is used as input of a Fully Connected Neural Network (FCNN) of two dense layers with the relu activation function. The class (hatefull or not) is obtained in an output layer with two units, relative to the number of classes, with the softmax activation function.

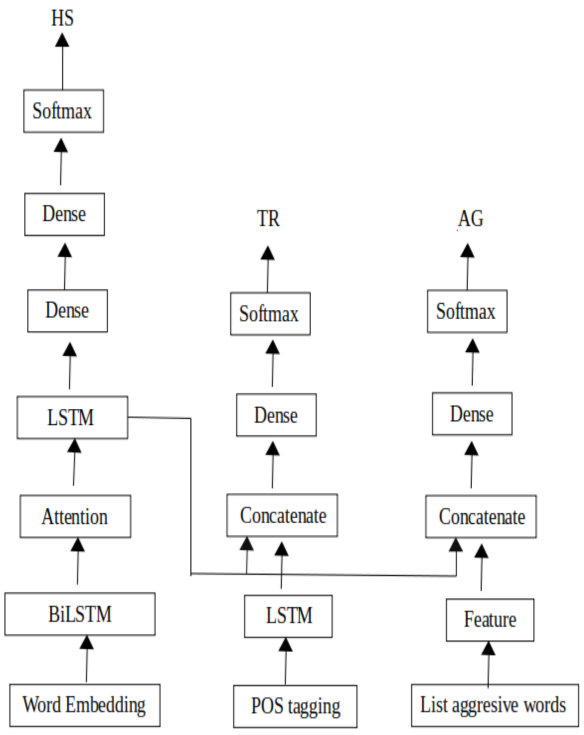


Figure 1: General architecture

2.2 Sub-task B

In the sub-task B the aim is to analyze some features of hateful messages, as discussed above. On the one hand, identifying against whom is directed the hate language (TR) and on the other hand, detecting whether a message with hate language is also aggressive.

For the task of detecting the target of hate (TR), the information of the part-of-speech tagging process of the tweets is used. The sequence of labels is analyzed with a LSTM RNN, obtaining a vector. Then, this vector is concatenated with R (from subtask A) and it is used as input to a dense layer with the relu activation function. Finally, the output layer has two neurons with the softmax activation function. In this way, the prediction corresponding to the offensive target in the tweets is obtained.

In the case of the task of classifying a hateful tweet in aggressive or not, a linguistic resource that consists of a dictionary of aggressive words is used. In this way, a linguistic characteristic is added to R according to the number of words in the tweet that appear in the dictionary. In addition, a one hot vector corresponding to the POS tags present in the tweet is added. With this new vector, the prediction is obtained in a similar way to the previous task, using another dense layer with the relu activation function.

3 Results

For the evaluation of the results of the task, different strategies and metrics are applied. For the sub-task A, systems are evaluated using standard evaluation metrics, including accuracy, precision, recall and F1-score. The submissions are ranked by F1-score. For the sub-task B, systems are evaluated with two criteria: partial match and exact match (EMR). In partial match, each dimension to be predicted (HS, TR and AG) is evaluated independently of the others using standard evaluation metrics including accuracy, precision, recall and F1-score, and then combined. In exact match, all the dimensions to be predicted are jointly considered. The submissions are ranked by the EMR measure.

Table 1 shows the results obtained for each of the languages in the sub-task A. In addition, the results of the system positioned in the first place of the ranking are shown for each of the metrics. In the same way, the results obtained for the sub-task B are shown in Table 2.

Language	Task A			
	Acc	P	R	F1-score
English	0.453	0.545	0.516	0.39
Best-English	0.653	0.69	0.679	0.651
Spanish	0.723	0.717	0.722	0.718
Best-Spanish	0.731	0.734	0.741	0.73

Table 1: Results for the sub-task A

Language	Task B	
	F1-score	EMR
English	0.532	0.268
Best-English	0.467	0.57
Spanish	0.74	0.618
Best-Spanish	0.755	0.705

Table 2: Results for the sub-task B

As can be seen, the results for English are very far from the best results obtained in the competition, reaching a low position in the ranking of the participating systems. On the other hand, the results for Spanish are better, reaching the eighth position in the ranking for task A and the eleventh for task B. However, these results are not good enough, which reveals that possibly the complexity of the model used is a problem. Therefore, a better approach might be to simplify the model in order to reduce the number of parameters to train. On the other hand, we think that it is very important for improving the results, to find discriminatory linguistic features, able of capturing the nature of the texts with hate; and to make a fined tunned of paramaters for each language.

4 Conclusion

In this paper we presented our solution for HatEval task in SemEval 2019. We used an approximation that combines a BiLSTM RNN with an attention mechanism to obtain a text representation vector. This vector is used as input in each of the models designed for each of the subtasks in which HatEval is divided. The results obtained were not very good, far away from the results obtained by the best system in the competition. As a conclusion of the analysis, it is believed that a better approximation would be to simplify the proposed model to reduce the number of parameters to train. Also, searching for discriminating linguistic features, which manage to capture the nature of texts with hate, should help to obtain better results.

References

Miguel Álvarez-Carmona, Estefanıa Guzmán-Falcón, Manuel Montes-y Gómez, Hugo Jair Escalante, Luis Villasenor-Pineda, Verónica Reyes-Meza, and Antonio Rico-Sulayes. 2018. Overview of mex-a3t at ibereval 2018: Authorship and aggressiveness analysis in mexican spanish tweets. In *Notebook Papers of 3rd SEPLN Workshop on Evaluation of Human Language Technologies for Iberian Languages (IBEREVAL), Seville, Spain*, volume 6.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Carlos Enrique Muniz Cuza, Gretel Liz De la Pena Sarracén, and Paolo Rosso. 2018. Attention mechanism for aggressive detection. In *CEUR Workshop Proceedings*, volume 2150, pages 114–118.

Elisabetta Fersini, Maria Anzovino, and Paolo Rosso. 2018a. Overview of the task on automatic misogyny identification at ibereval. In *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018), co-located with 34th Conference of the Spanish Society for Natural Language Processing (SEPLN 2018). CEUR Workshop Proceedings. CEUR-WS. org, Seville, Spain*.

Elisabetta Fersini, Debora Nozza, and Paolo Rosso. 2018b. Overview of the evalita 2018 task on automatic misogyny identification (ami). *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA18), Turin, Italy. CEUR. org*.

Ritesh Kumar, Atul Kr Ojha, Marcos Zampieri, and Shervin Malmasi. 2018. Proceedings of the first workshop on trolling, aggression and cyberbullying (trac-2018). In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*.

Gretel Liz De la Peña Sarracén, Reynaldo Gil Pons, Carlos Enrique Muñiz-Cuza, and Paolo Rosso. 2018. Hate speech detection using attention-based LSTM. In *Proceedings of the Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018) co-located with the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018), Turin, Italy, December 12-13, 2018*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Zeerak Waseem, Wendy Hui Kyong Chung, Dirk Hovy, and Joel Tetreault. 2017. Proceedings of the first workshop on abusive language online. In *Proceedings of the First Workshop on Abusive Language Online*.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the germeval 2018 shared task on the identification of offensive language. In *14th Conference on Natural Language Processing KONVENS 2018*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

INF-HatEval at SemEval-2019 Task 5: Convolutional Neural Networks for Hate Speech Detection Against Women and Immigrants on Twitter

Alison P. Ribeiro
Institute of Informatics
Federal University of Goiás
Goiânia – Goiás – Brazil
alisonrib17@gmail.com

Nádia F. F. da Silva
Institute of Informatics
Federal University of Goiás
Goiânia – Goiás – Brazil
nadia@inf.ufg.br

Abstract

In this paper, we describe our approach to detect hate speech against women and immigrants on Twitter in a multilingual context, English and Spanish. This challenge was proposed by the SemEval-2019 Task 5, where participants should develop models for hate speech detection, a two-class classification where systems have to predict whether a tweet in English or in Spanish with a given target (women or immigrants) is hateful or not hateful (Task A), and whether the hate speech is directed at a specific person or a group of individuals (Task B). For this, we implemented a Convolutional Neural Networks (CNN) using pre-trained word embeddings (GloVe and FastText) with 300 dimensions. Our proposed model obtained in Task A 0.488 and 0.696 F1-score for English and Spanish, respectively. For Task B, the CNN obtained 0.297 and 0.430 EMR for English and Spanish, respectively.

1 Introduction

With the growth of users in social networks, there was also an increase in the odious activities that permeate these communicative structures. According to Nockleby et al. (2000), hate speech can be defined as any communication that deprecates a person or a group based on some characteristics such as race, color, ethnicity, gender, nationality, religion or other features. And the main motive that encourages users to spread hate on social networks is anonymity, so users can spread hate words to a particular target. For this reason, the hatred propagated can generate irreversible consequences, where young people who approach with cyberbullying and homophobia, mainly, commit suicide.

Nowadays, social networks like Twitter[1], Facebook[2] and YouTube[3] are pressured to develop tools to fight the proliferation of hate in their networks. A good example of this is the German government that threatened to fine social networks by up to 50 million euros if they did not fight the spread of hate (Gambäck and Sikdar, 2017).

However, while there is plenty of available content on social networks, the task of detecting hate speech remains difficult, largely because of the use of different sets of data for work, lack of benchmarking, and efficient approaches. Waseem, for example, bring a study focused on the detection of racism and sexism, whereas Nahar et al. 2012 and Sanchez and Kumar 2011) conducted a survey on detecting bullying. For the detection of homophobia, misogyny and xenophobia, the number of papers is still limited, one can cite a recent paper (Sanguinetti et al., 2018), where the authors sought to identify hate speech against immigrants. However, it is important that new research is publicized, because only in this way will it be possible to fight against hate in social networks.

Introducing a brief definition of hate speech and the importance of combating it, SemEval-2019 proposed a task in which it challenges participants to develop systems for detecting hate speech against women and immigrants on Twitter from a multilingual perspective , for English and Spanish.

The task was articulated around two related subtasks for each one of the languages involved: a basic task about hate speech, and another where refined hate content resources will be investigated to understand how existing approaches can handle the identification of especially dangerous forms of hatred, that is, those in which incitement is directed against an individual rather than against a group of people, and where aggressive behavior of

[1]https://twitter.com/

[2]https://www.facebook.com/

[3]https://www.youtube.com/

the perpetrator can be identified as a prominent feature of the expression of hatred. In order to reach this goal, this work proposed to develop a Convolutional Neural Network with the use of word embeddings.

The paper is organised as follows: previous work on hate-speech identification is discussed in Section 2. Section 3 presents details about the task, data sets and evaluation methods. Section 4 describes the methodology for categorizing hate speech based on deep learning, while experiments and results are reported in Section 5. Finally, Section 6 summarises the discussion.

2 Related Work

Some computational methods to detect hate speech are presented in this section. An example is the work of Badjatiya et al. (2017) that applied several algorithms of machine learning and deep learning, among them: Logistic Regression, Support Vector Machine, Random Forest, Gradient Boosted Decision (GBDT), CNN and Long Short-Term Memory (LSTM). As a baseline they used char n-grams and bag-of-words, and as word embeddings they used GloVe and FastText. The objective was to classify if a tweet is racist, sexist or none, and the best result was 0.930 of F1-score, which was obtained through a LSTM model with Random Embedding and GBDT.

Another work that also follows a line of ternary classification was proposed by Malmasi and Zampieri (2017), where the purpose is to classify a tweet as hateful, offensive (but not hateful) or offensive language. For this, the researchers proposed an approach based on n-grams and word skip-grams using Support Vector Machine with cross-validation, the best result achieved was 0.78 of accuracy.

Gambäck and Sikdar (2017) developed a Convolutional Neural Network to classify hate speech on Twitter. In this case, the authors used 4 categories: racism, sexism, both (racism and sexism) and non-hate-speech. The structure of CNN was constructed with convolutional layers and pooling of 4 modes: character 4-grams, word vector based on semantic information built using word2vec (Mikolov et al., 2013a), randomly generated word vectors and word embeddings with character n-grams. In the classification phase, the softmax function and cross-validation with 10-folds were applied, the model based on word2vec embeddings best performed with 0.783 of F-score.

A recent study developed by Gaydhani et al. (2018) sought to address the difference between offensive language and hate speech, then the authors proposed several machine learning models based on n-grams and TF-IDF. The models were analyzed considering several n-values in n-grams and TF-IDF normalization methods. Consequently, the best result among several approaches was 0.956 of accuracy.

3 SemEval-2019 Task 5

In this section we will describe some details about data sets, tasks, and evaluation methods.

3.1 Dataset

The data for the task consists of 9000 tweets in English for training, 1000 for develop and 2805 for test. For Spanish, 4469 tweets for training, 500 for develop and 415 for test. The data were structured in 5 columns: id, text, Hate Speech (HS), Target Range (TR) and Aggressiveness (AG). See an example in the Table 1 (Basile et al., 2019). If the $@username$ is a woman, we have a case of feminicide.

id	text	HS	TR	AG
93874	@username stupid wish you die.	1	1	1
18267	Leftwing filth Deport them all. #Sendthemback	1	0	1
18345	1,500 migrants have died in Mediterranean in 2018	0	0	0

Table 1: Example of hate speech. Some examples are also taken from the data.

3.2 Task A

The task A is a two-class classification problem in which participants have to predict whether a tweet, in English or Spanish, with a particular target (women or immigrants) is hateful or not hateful – Hate Speech (1/0).

3.3 Task B

The purpose of this task is to: (i) classify hate tweets into English and Spanish, where tweets with hate speech, against women or immigrants, were identified as aggressive or non-aggressive, and (ii) identify the harassed target as just one person or group of individuals.

3.4 Evaluation

For the results evaluation of both tasks A and B, different metrics were used in order to allow more refined conclusions.

Task A. The systems will be evaluated according to the following metrics: accuracy, precision, recall and F1-score. The equations below show how the calculations are done. In the case of this task, the scores will be classified by F1-score. For better understanding, we will show the following definitions:

- *True positive* (TP): means a correct classification as odious. For example, the royal class is hateful and the model ranks as hateful.
- *True negative* (TN): means a correct classification as not hateful. For example, the royal class is not hateful and the model ranked as not hateful.
- *False positive* (FP): means a wrong classification as odious. For example, the royal class is not hateful and the model rated it as hateful.
- *False negative* (FN): means a wrong classification not hateful. For example, the royal class is hateful and the model ranked as not hateful.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$F1\text{-}score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (4)$$

Task B. In this task, the evaluation metrics are two: partial match and exact match. The strategy for the partial match is to evaluate the Hate Speech, Target Range and Agressiveness classes independently of each other using the metrics defined above. However, each system will include all measures and a summary of the performance in terms of macro-average F1-score, calculated according to the Equation 5. The exact match considers the predicted classes together, thus computing the Exact Match Ratio (Kazawa et al., 2005). Given the set of data consisting of n multi-label samples (Xi, Yi), where Xi denotes the i-th instance and Yi corresponds to the labels to be predicted (HS, TR and AG), the Exact Match Ratio (EMR) is calculated according to Equation 6.

$$F1\text{-}score = \frac{F_1(HS) + F_1(AG) + F_1(TR)}{3} \quad (5)$$

$$EMR = \frac{1}{n}\sum_{i=1}^{n} I(Yi, Zi) \quad (6)$$

where Zi denotes the set of labels predicted for the i-th instance and I is the indicator function.

4 Methodology

In this section, we describe the details of our proposed methods, including data preprocessing, neural networks and word embeddings.

4.1 Preprocessing

This step consists in eliminating noises and terms that have no semantic significance in classes prediction. For this, we performed the removal of links, numbers, special characters, and *stop words* (words with low discriminative power, for example, "is", "that" etc.) and standardized in lowercase.

4.2 Word embeddings

Word Embeddings (Bengio et al., 2003) is a supervised statistical language model trained using deep neural networks. The purpose of this language model is to predict the next word, given the previous words of the sentence. The vector embeddings was a great advance in relation to the strategies based on the bag-of-words, which justifies its use in several works (Nakov et al., 2016; Poria et al., 2015; Cliche, 2017; Zhou et al., 2018; Rotim et al., 2017). For the proposed task, we use the GloVe (Pennington et al., 2014) and FastText (Joulin et al., 2016) model with 300 dimensions.

For the English language, we made use of the Stanford pre-trained GloVe (Pennington et al., 2014) where word embedding were trained with Wikipedia 2014 and Gigaword 5, while FastText (Joulin et al., 2016) was trained in Wikipedia 2017, UMBC webbase corpus and statmt.org news.

For the Spanish language, the GloVe vocabulary was computed from SBWC (Pennington et al., 2014; Cardellino, 2016), while FastText was computed from the Spanish Wikipedia (Bojanowski et al., 2016).

4.3 Convolutional Neural Networks

Initially, the Convolutional Neural Network architecture was designed for image processing, however it has been commonly used in the sentiment analysis (Wang et al., 2016; Cambria et al., 2016; Rosenthal et al., 2017; dos Santos and Gatti, 2014; Poria et al., 2015).

For the purpose of the task, a CNN was implemented based on the architecture proposed by (Zhang and Wallace, 2015), and this implementation can be divided in two steps: feature extraction and classification. In the feature extraction step, only two layers of convolution and two layers of pooling were used, with tanh activation function. Four filters were used, 2 of 3 dimensions and 2 of 4 dimensions. Each filter refers to the classic n-grams technique (extremely used in bag-of-words based models), which consists of processing a group of n words, in order to consider not only isolated words in a tweet, but also the context in which they are inserted. The filters are applied under the vector representation of the input tweets (embedding layer), using the concept of Back propagation to adjust the weights dynamically. According to Poria et al. (2016), these filters can extract lexical, syntactic or semantic features automatically. Finally, the two layers of convolution and pooling are concatenated and directed to the next step.

In the classification stage, we used two dense layers, the first one has 512 neurons, relu function of activation and dropout of 0.5. The second has a neuron and sigmoid activation function, phase where classification occurs. For the training, the loss and optimization functions used were binary_crossentropy and RMSprop (Hinton et al., 2012) (with learning rate 0.001), respectively.

5 Results

In this section we will discuss the results obtained by using a CNN for the detection of hate speech and the target of hate.

Task A				
English				
Model	**F1**	**P**	**R**	**Acc**
CNN-FastText	0.488	0.628	0.574	0.520
Spanish				
Model	**F1**	**P**	**R**	**Acc**
CNN-GloVe	0.696	0.708	0.712	0.696

Table 2: Results obtained related to Task A.

We obtained 0.488 of F1-score for English and 0.696 for Spanish with our CNN model using word embeddings, as shown in Table 2. This result also suggests that the combination of CNN and GloVe provides better results for this task.

English			
Class	**F1**	**P**	**R**
hateful	0.617	0.465	0.916
not hateful	0.359	0.792	0.232
Spanish			
Class	**F1**	**P**	**R**
hateful	0.685	0.598	0.802
not hateful	0.707	0.817	0.622

Table 3: Confusion matrix concerning task A.

The Table 3 displays the results of F1-score, Precision and Recall reached by class for each language. The F1-score can be used to measure the performance of the classifier, in this case CNN ranked the hateful class better, obtaining 0.617 of F1-score, while the result for not hateful class was 0.359 of F1-score in the English language.

From the perspective of the Spanish language, CNN obtained good results in the classification of both classes, hateful and not hateful, with 0.685 F1-score and 0.707 F1-score, respectively.

Task B		
English		
Model	**F1**	**EMR**
CNN-FastText	0.577	0.297
Spanish		
Model	**F1**	**EMR**
CNN-FastText	0.609	0.430

Table 4: Results obtained related to Task B.

Recapitulating the idea of task B, where the goal is to identify the target of the hate speech, that is, whether it is a single person or a group of individuals. Knowing that there is hate speech in the tweet (HS is 1), then one must detect if the target is only one person (TR is 1) or if it is a group of individuals (TR is 0), and if there is presence of aggressiveness in speech (AG is 1) or not (AG is 0). In this case, the EMR measure shows a percentage in which it corresponds to an accuracy rate, that is, it measures how much the model has managed not only to classify the hate speech, but also the target and the aggressiveness. The Table 4 shows the results of task B, where it was possible to obtain 0.297 EMR for English and 0.430 EMR for Spanish.

6 Conclusion

In this paper, we introduced the system that we proposed for SemEval-2019, task 5. Our goal was to experience an architecture that was adapted from a CNN using word embeddings. The task was to detect hate speech against women and immigrants on Twitter from a multilingual perspective, English and Spanish. We participate in two subtasks directed to the two languages, and we obtain the 18th position in the ranking of task A and the 19th position of task B in the English language. In the Spanish language, we obtain the 24th position in the ranking for both tasks.

The success of deep learning depends on finding an architecture to fit the task. Furthermore, as deep learning has scaled up to more challenging tasks, the architectures have become difficult to design by hand. In this paper, a CNN was implemented based on the architecture proposed by Zhang and Wallace 2015 and a fine-tuning of hyperparameters was not done for the proposed tasks (tasks A and B). In addition, other features were not exploited as sarcasm and irony, inherent in this type of domain. We intend to explore these and other features in future work.

Another discussion can be raised regarding the best performance to have happened in Spanish. The main hypothesis is related to the nature of the corpus used. It is observed that the test set of Spanish is smaller than that of English, besides being a corpus with "simpler texts to be classified" (Spanish texts have few signs of sarcasm). Such analyzes need further studies and will be evaluated in future work.

For future work as well, it would be interesting to explore systems that use different parameters for CNN and other word embeddings, such as Word2Vec (Mikolov et al., 2013b). It would also be interesting to construct an LSTM with attention mechanism proposed by Lin et al. (2017) and compare its performance.

References

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019).* Association for Computational Linguistics", location = "Minneapolis, Minnesota.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Erik Cambria, Soujanya Poria, Rajiv Bajpai, and Björn Schuller. 2016. Senticnet 4: A semantic resource for sentiment analysis based on conceptual primitives. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2666–2677.

Cristian Cardellino. 2016. Spanish Billion Words Corpus and Embeddings.

Mathieu Cliche. 2017. Bb_twtr at semeval-2017 task 4: Twitter sentiment analysis with cnns and lstms. *arXiv preprint arXiv:1704.06125*.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using convolutional neural networks to classify hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Aditya Gaydhani, Vikrant Doma, Shrikant Kendre, and Laxmi Bhagwat. 2018. Detecting hate speech and offensive language on twitter using machine learning: An n-gram and tfidf based approach. *arXiv preprint arXiv:1809.08651*.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. 2012. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *CoRR*, abs/1607.01759.

Hideto Kazawa, Tomonori Izumitani, Hirotoshi Taira, and Eisaku Maeda. 2005. Maximal margin labeling for multi-topic text categorization. In *Advances in neural information processing systems*, pages 649–656.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *CoRR*, abs/1703.03130.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting hate speech in social media. *CoRR*, abs/1712.06427.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Vinita Nahar, Sayan Unankard, Xue Li, and Chaoyi Pang. 2012. Sentiment analysis for effective detection of cyber bullying. In *Asia-Pacific Web Conference*, pages 767–774. Springer.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 task 4: Sentiment analysis in twitter. In *Proceedings of the 10th international workshop on semantic evaluation (semeval-2016)*, pages 1–18.

John T. Nockleby, Kenneth L. Karst Leonard W. Levy, and Adam Winkler. 2000. *Hate Speech. In Encyclopedia of the American Constitution*. New York : Macmillan Reference USA, ©2000.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Soujanya Poria, Erik Cambria, and Alexander Gelbukh. 2015. Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 2539–2544.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, and Prateek Vij. 2016. A deeper look into sarcastic tweets using deep convolutional neural networks. *arXiv preprint arXiv:1610.08815*.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518.

Leon Rotim, Martin Tutek, and Jan Šnajder. 2017. Takelab at semeval-2017 task 5: Linear aggregation of word embeddings for fine-grained sentiment analysis of financial news. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 866–871.

Huascar Sanchez and Shreyas Kumar. 2011. Twitter bullying detection. *ser. NSDI*, 12:15–15.

Manuela Sanguinetti, Fabio Poletto, Cristina Bosco, Viviana Patti, and Marco Stranisci. 2018. An italian twitter corpus of hate speech against immigrants. In *Proceedings of LREC*.

Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78.

Jin Wang, Liang-Chih Yu, K Robert Lai, and Xuejie Zhang. 2016. Dimensional sentiment analysis using a regional cnn-lstm model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 225–230.

Zeerak Waseem. 2016. Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. In *Proceedings of the first workshop on NLP and computational social science*, pages 138–142.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

Liyuan Zhou, Qiongkai Xu, Hanna Suominen, and Tom Gedeon. 2018. Epution at semeval-2018 task 2: Emoji prediction with user adaption. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 449–453.

JCTDHS at SemEval-2019 Task 5: Detection of Hate Speech in Tweets using Deep Learning Methods, Character N-gram Features, and Preprocessing Methods

Yaakov HaCohen-Kerner, Elyashiv Shayovitz,
Shalom Rochman, Eli Cahn, Gal Didi, and Ziv Ben-David
Department of Computer Science, Jerusalem College of Technology, Lev Academic Center
21 Havaad Haleumi St., P.O.B. 16031, 9116001 Jerusalem, Israel
kerner@jct.ac.il, elyashiv12@gmail.com,
shal.rochman@gmail.com, eli.cahn@gmail.com,
galdd8@gmail.com, and benda1237@gmail.com

Abstract

In this paper, we describe our submissions to SemEval-2019 contest. We tackled subtask A - "a binary classification where systems have to predict whether a tweet with a given target (women or immigrants) is hateful or not hateful", a part of task 5 "Multilingual detection of hate speech against immigrants and women in Twitter (HatEval)". Our system JCTDHS (Jerusalem College of Technology Detects Hate Speech) was developed for tweets written in English. We applied various supervised ML methods, various combinations of n-gram features using the TF-IDF scheme. In addition, we applied various combinations of eight basic preprocessing methods. Our best submission was a special bidirectional RNN, which was ranked at the 11th position out of 68 submissions.

1 Introduction

Hate Speech is usually defined as communication that contains contempt or hatred towards a person or a group of people on the basis of some characteristic e.g., color, ethnicity, gender, nationality, race, religion, and sexual orientation.

The phenomenon of hate speech in social media has grown in recent years (Eadicicco. 2014). A strong connection between hate speech and actual hate crimes has been shown in Watch (2014). In light of the huge amount of information in social media, early detection of people using hate speech could prevent them from carrying out their hate speech. Therefore, the detection of hate speech in social media has become an issue of increasing importance (Moulson. 2016).

In this paper, we describe our six models (each model with another team member as the first author) submitted to task 5-A for tweets written in English. The full description of this task is given in Basile et al. (2019).

The structure of the rest of the paper is as follows. Section 2 introduces a background concerning hate speech, tweet classification, and data preprocessing. Section 3 presents, in general, the description of Task 5. In Section 4, we describe the submitted models and their experimental results. Section 6 summarizes and suggests ideas for future research.

2 Background

2.1 Hate Speech

Waseem and Hovy (2016) introduced a list of criteria founded in critical race theory and used them to label a publicly available corpus of more than 16k tweets with tags about both racial and sexist offenses. Nobata et al. (2016) developed a machine learning based method to detect hate speech on online user comments from two domains. They also built a corpus of user comments annotated accordingly to three subcategories (hate speech, derogatory, profanity). Schmidt and Wiegand (2017) introduced a survey of the NLP methods that were developed in order to detect hate speech. Davidson et al. (2017) presented a multi-class classifier to distinguish between three categories: hate speech, offensive language, and none of these two. The analysis of the predictions and the errors shows when they can reliably separate hate speech from other types of offensive language (e.g., tweets with the highest predicted probabilities of being hate speech tend to contain multiple racial or homophobic slurs) and when this differentiation is more difficult (e.g., many tweets misclassified as hate speech contain terms that can be considered racist and sexist; however it is apparent that many Twitter users use this type of language in their everyday communications). Anzovino et al. (2018) built a labeled corpus containing 2,227 misogynous (hate speech against women) tweets and no-misogynous tweets and explored various NLP features and ML models for detecting and classifying misogynistic language.

2.2 Tweet Classification

Tweet classification is the task to automatically classify a tweet into one of a set of predefined classes. This research domain has been growing rapidly in recent years. Twitter as one of the leading social networks presents challenges to the researchers since tweets are informal, short, and contain various misspellings, shortenings, and slang words (HaCohen-Kerner et al., 2017).

2.3 Data preprocessing

Data preprocessing is an important step in data mining (DM) and ML processes. In data files, it is common to find typos, emojis, slang, HTML tags, spelling mistakes, irrelevant and redundant information. Analyzing data that has not been carefully cleaned or pre-processed might lead to misleading results.

Not all of the preprocessing types are considered effective by all text classification (TC) researchers. For instance, Forman (2003), in his study on feature selection metrics for TC, claimed that stop words frequently occur and are ambiguous and therefore should be removed, However, HaCohen-Kerner et al. (2008) demonstrated that the use of word unigrams including stop words lead to improved TC results compared to the results obtained using word unigrams excluding stop words in the domain of Hebrew-Aramaic Jewish law documents.

In our system, we applied various combinations of eight basic preprocessing types: C - spelling Correction (The spelling correction is performed using an autocorrect library, written by McCallum (2014)[1]), H – HTML Tags Removal, L – converting to Lowercase letters, P – Punctuation removal, S – Stopwords Removal, R – Repeated characters removal (repeated characters were removed and only one character was left), T – sTemming, and M - leMmatizion) in order to check whether they improve TC or not.

3 Task Description

Task 5 deals with two tasks related to hate speech detection in Twitter with two specific targets, women and immigrants, for tweets in English and Spanish. We participated only in Task 5-A for tweets written in English, i.e., a two-class classification task where we have to predict whether a tweet with a given target (women or immigrants) is hateful or not hateful.

The datasets of Task 5-A consists of Train, Dev and Test datasets. The Train dataset contains 9,000 categorized tweets: 3,783 HS (hateful speech) tweets and 5,217 NHS (not hateful speech) tweets. The Dev dataset first published without labels, and they were added only in the final evaluation phase. The Dev set contains 1,000 tweets: 427 HS tweets, and 572 NHS tweets. The Test dataset contains 3,000 uncategorized Tweets.

4 The Submitted Models and Experimental Results

We have submitted six models (one for each author) to task 5-A for tweets written in English. The general TC algorithm is as follows.

1. Using a TF-IDF scheme, find the optimal number of word n-grams and combination of pre-processing types.
2. Apply various supervised ML methods including RNN models and others to find the best accuracy results.

Table 1 presents the main characteristics and results of our six submitted models in descending order according to their F1-Macro (F-M) score on the test set. Figure 1 presents our final RNN model using n-gram features in layer N.

[1] https://github.com/phatpiglet/autocorrect/ Last access date: 19-MAR-19.

The first name of the model authors	Pre-processing	Model			Score		
		RNN Architecture	N-gram Features	Its Fully Connected Layer uses	CV Score (F- M)	Test Score (F- M)	Rank
galdd8@ gmail.com	CHLPRS	Bidirectional RNN with 4 hidden layers. Each layer contains128 LSTM units and Dropout layer (0.4). GloVe of 200d used for embedding	100 char trigrams, no skips	Logistic Regression	0.737	0.5	11 \ 68
elyashiv12 @ gmail.com	None	RNN contains 128 LSTM units, and Dropout layer (0.3). GloVe of 200d used for embedding	None	None	0.751	0.49	15 \ 68
kerner@ jct.ac.il	None	Bidirectional RNN with 4 hidden layers. Each layer contains128 LSTM units and Dropout layer (0.4). GloVe of 200d used for embedding	None	None	0.754	0.48	21 \ 68
ShalomRo chman	CHLPRS	Bidirectional RNN with 4 hidden layers. Each layer contains128 LSTM units and Dropout layer (0.4). GloVe of 200d used for embedding	200 char bigrams, no skips	SVM (SGD Variant)	0.749	0.42	42 \ 68
benda1237 @ gmail.com	CHLPRS	Bidirectional RNN with 4 hidden layers. Each layer contains128 LSTM units and Dropout layer (0.4). GloVe of 200d used for embedding	200 word unigram, no skips	SVM (SGD Variant)	0.713	0.41	41 \ 68
ecahn	CHLPRS	Bidirectional RNN with 2 hidden layers. Each layer contains128 LSTM units and Dropout layer (0.4). GloVe of 200d used for embedding	300 char bigrams, no skips	SVM (SVC Variant)	0.759	0.38	54 \ 68

Table 1: Results of our 6 models in task-A.

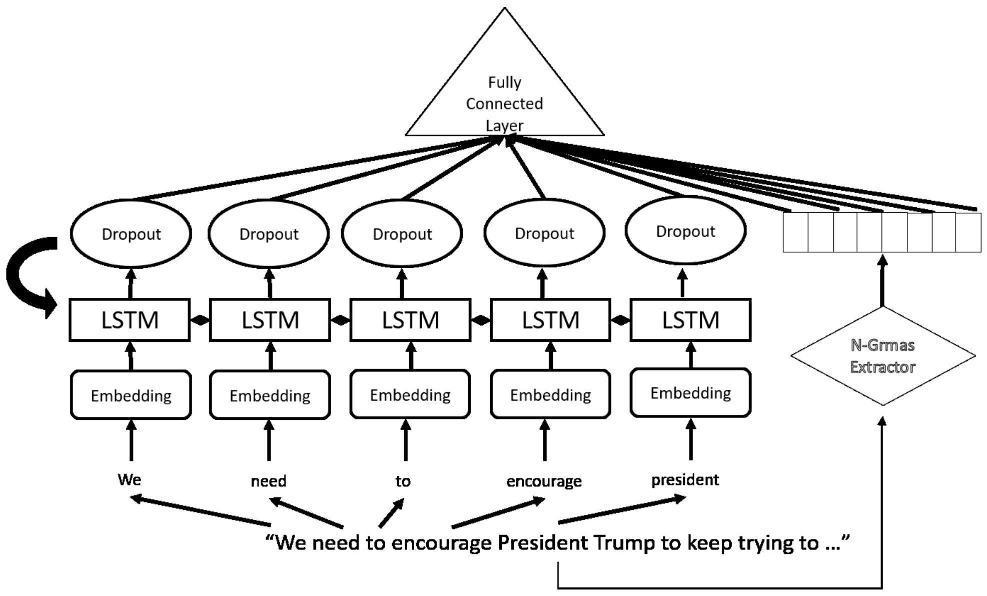


Figure 1: Final RNN model with n-gram features in layer N.

Analysis of the results presented in Table 1 shows that our best Macro F-measure score (as opposed to F1 of hate speech alone) for the test set (0.5) was obtained by a bidirectional RNN with 4 hidden layers, 128 LSTMs, 0.4 Dropout and a GloVe (Pennington et al., 2014) of 200d special for Twitter as an embedded layer. The best combination of pre-processing types was found to be CHLPRS, which means to activate all pre-processing types.

In our experiments, we used the following framework Python 3.6 with Keras[2] and Scikit-Learn in PyCharm IDE (Pedregosa et al., 2011) using the TF-IDF scheme called TfidfTransformer[3]). The accuracy of each ML model was estimated by a 5-fold cross-validation testing. The vocabulary words were used as zero-vectors during the word-to-embedding conversion. The Fully connected layer (FC) is the last layer in RNN models. It performs the final classification. The activation function of the FC layer is the sigmoid function. We used the RMSProp optimizer and 30 epochs for each model.

5 Conclusions and Future Research

In this paper, we describe our submissions to Task 5-A of SemEval-2019 contest. Our system JCTDHS (Jerusalem College of Technology Detects Hate Speech) was developed for tweets written in English. We used a TF-IDF scheme and we performed various combinations of six pre-processing methods to improve the performance.

Our best submission for Task 5-A was a bidirectional RNN with 4 hidden layers while each layer contains128 LSTM units, a dropout layer (0.4), and a GloVe (Global Vectors for Word Representation) of 200d that was used for embedding. GloVe was developed by the Stanford NLP Group (Pennington et al., 2014). It applies a co-occurrence matrix and by using matrix factorization.

This submission was ranked at the 11th out of 68 submissions for tweets written in English.

Future research proposals are as follows. It is known that many tweets include acronyms (abbreviations) that are presented in different forms. Acronym disambiguation (HaCohen-Kerner et al., 2010A), i.e., selecting the correct long form of the acronym depending on its context will enrich the tweet's text and will enable better TC.

More ideas that may contribute to better classification are implementing TC using (1) additional feature sets such as stylistic feature sets (HaCohen-Kerner et al., 2010B) and key phrases that can be extracted from the text files (HaCohen-Kerner et al., 2007) and (2) additional deep learning models.

[2] https://github.com/gucci-j/vae.
[3] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html#sklearn.feature_extraction.text.TfidfTransformer

Acknowledgments

This research was partially funded by the Jerusalem College of Technology, Lev Academic Center.

References

Maria Anzovino, Elisabetta Fersini, and Paolo Rosso. 2018. Automatic Identification and Classification of Misogynistic Language on Twitter. International Conference on Applications of Natural Language to Information Systems. Springer, Cham.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women on Twitter. Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), Association for Computational Linguistics, Minneapolis, Minnesota, USA.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In: Proceedings of the 12th International AAAI Conference on Web and Social Media.

Lisa Eadicicco. 2014. This female game developer was harassed so severely on twitter she had to leave her home. Business Insider, 12(10).

George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. Journal of machine learning research, 3(Mar), 1289-1305.

Yaakov HaCohen-Kerner, Ittay Stern, David Korkus, and Erick Fredj. 2007. Automatic machine learning of keyphrase extraction from short HTML documents written in Hebrew. *Cybernetics and Systems: An International Journal*, *38*(1), 1-21.

Yaakov HaCohen-Kerner, Dror Mughaz, Hananya Beck, and Elchai Yehudai 2008. Words as classifiers of documents according to their historical period and the ethnic origin of their authors. Cybernetics and Systems: An International Journal, 39(3), 213-228.

Yaakov HaCohen-Kerner, Ariel Kass, and Ariel Peretz. 2010A. HAADS: A Hebrew Aramaic abbreviation disambiguation system. Journal of the American Society for Information Science and Technology, 61(9), 1923-1932.

Yaakov HaCohen-Kerner, Hananya Beck, Elchai Yehudai, and Dror Mughaz. 2010B. Stylistic feature sets as classifiers of documents according to their historical period and ethnic origin. Applied Artificial Intelligence, 24(9), 847-862.

Yaakov HaCohen-Kerner, Ziv Ido, and Ronen Ya'akobov. 2017. Stance classification of tweets using skip char Ngrams. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 266-278). Springer, Cham.

Hate Speech Watch. 2014. Hate crimes: Consequences of hate speech. http: //www.nohatespeechmovement. org/hate-speech-watch/focus/ consequences-of-hate-speech, June. Seen on 23rd Jan. 2016.

Jonas McCallum. 2014. Python 3 Spelling Corrector. https://pypi.python.org/pypi/autocorrect/0.1.0.

Geir Moulson. 2016. Zuckerberg in Germany: No place for hate speech on Facebook. http://abcnews.go.com/Technology/ wireStoryZuckerberg-place-hatespeech-facebook-37217309. Accessed 10/03/2016.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In: Proceedings of the 25th International Conference on World Wide Web, pp. 145–153. International World Wide Web Conferences Steering Committee.

Fabian Pedregosa, Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R. and Dubourg, V. and Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), 2825-2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

Anna, Schmidt, and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In: Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics, Valencia, Spain, pp. 1–10.

Zeerak Waseem, and Dirk Hov. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on Twitter. In: SRW@ HLT-NAACL, pp. 88–93.

Know-Center at SemEval-2019 Task 5: Multilingual Hate Speech Detection on Twitter using CNNs

Kevin Winter
Know-Center GmbH
Inffeldgasse 13
Graz, 8010, Austria
kwinter@know-center.at

Roman Kern
Know-Center GmbH
Inffeldgasse 13
Graz, 8010, Austria
rkern@know-center.at

Abstract

This paper presents the *Know-Center* system submitted for task 5 of the SemEval-2019 workshop. Given a Twitter message in either English or Spanish, the task is to first detect whether it contains hateful speech and second, to determine the target and level of aggression used. For this purpose our system utilizes word embeddings and a neural network architecture, consisting of both dilated and traditional convolution layers. We achieved average F1-scores of 0.57 and 0.74 for English and Spanish respectively.

1 Introduction

The ever-increasing number of message board forums, social media platforms and other websites that allow user comments enable participants to express their opinions freely and sometimes even anonymously. This barley restricted access in combination with the unmanageably vast amount of user-generated content unfortunately also creates an environment, which is vulnerable to profanity and hateful speech, rendering it hostile to the individuals or groups of people targeted. This problem is of increasing importance (Kettrey and Laster, 2014) and calls to establish systems that allow for automated detection of such behavior.

While detecting abusive language by it self is already a challenging task, (Malmasi and Zampieri, 2018) showed that it is even harder to differentiate between its subtypes. Messages containing profanity, sexism, racism and other forms of hateful speech may be formed using very similar vocabularies. Additionally, these subtypes may overlap, making data sets dependent on the subjective judgments of annotators. This may be particularly true for finding a threshold distinguishing aggressive and none-aggressive speech. In a task organized as part of COLING 2018 (Kumar et al., 2018) 15,000 participants were asked to detect aggressive behaviour in a data set of Facebook posts. Even the best system only obtained a weighted F1-score of 0.64.

Task 5 of the SemEval-2019 workshop (Basile et al., 2019) aims to accelerate the research and development of such systems. It comprises two distinct subtasks. The first subtask is devoted to detect hateful speech against immigrants and women in Twitter messages. For the second subtask, messages that are detected to be hateful are investigated further. Here, the goal is to identify aggressive behaviour and the target, women or immigrant, harassed. Given the multilingual nature of the task, systems for both English and Spanish are required. For the design and evaluation of these systems, training data sets for both languages are provided. The English data set consists of 9000 annotated messages whereas the Spanish data set consists of 4500.

Closely related, another task is hosted at the SemEval-2019 workshop, dealing with offensive language in social media and the individuals and groups targeted by it (Zampieri et al., 2019).

The rest of the paper is organized as follows. In the next section we give an overview of work that has been done in the field of detecting hateful and abusive speech. Section 3 describes our submitted system, including the pre-processing performed as well as the classifier trained on the data sets provided. In section 4 we show the results of our system and compare them with those of other participants in this challenge. Section 5 contains a brief summary.

2 Related Work

The problem of abusive behaviour in online media has been addressed in various fields. (Olteanu et al., 2018) address the causes for hateful speech

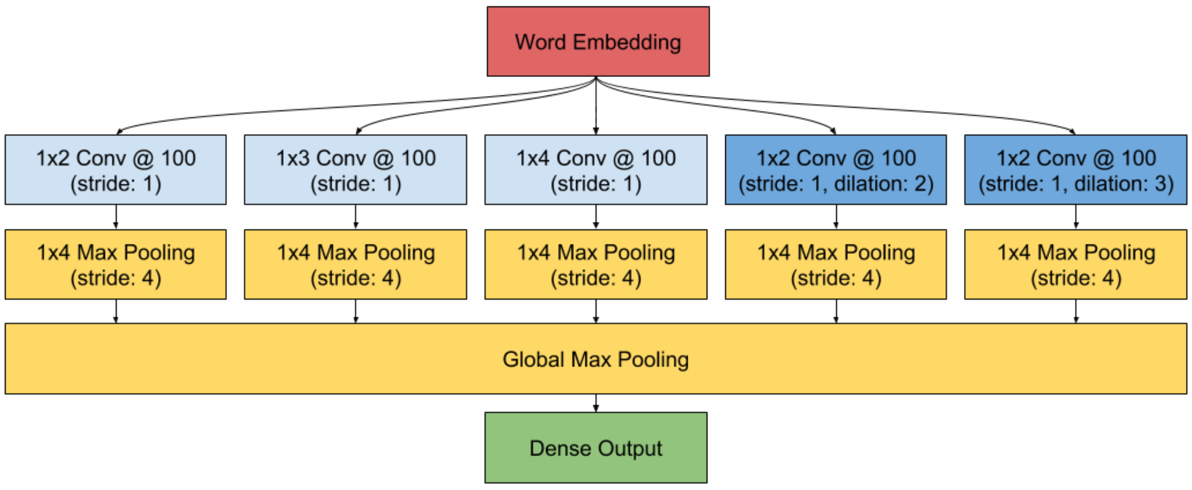

Word Embedding
1x2 Conv @ 100 (stride: 1)
1x3 Conv @ 100 (stride: 1)
1x4 Conv @ 100 (stride: 1)
1x2 Conv @ 100 (stride: 1, dilation: 2)
1x2 Conv @ 100 (stride: 1, dilation: 3)
1x4 Max Pooling (stride: 4)
Global Max Pooling
Dense Output

Figure 1: Network architecture of our classifier. Our system makes use of different strategies of applying convolutions, including dilated convolutions.

and in particular the effects of violent attacks performed by extremist groups and individuals. They were able to show, that such events majorly fuel hateful comments on Twitter and Reddit.

For the detection and classification task multiple methods have been previously explored. One approach involves the identification of words and parts of words in the form of character n-grams, that are the most indicative of hateful speech (Waseem and Hovy, 2016; Nobata et al., 2016). A potential downside of such approach might be that the meaning of a word may depend on the context it is used in (Sood et al., 2012). Chen et al. (2012) tried to overcome this issue by employing word n-grams. However, this is associated with an increase in feature space.

Another approach is to employ word embeddings in order to capture similarities between words (Badjatiya et al., 2017). Extending this method, Djuric et al. (2015) used paragraph2vec (Le and Mikolov, 2014) to encode whole messages and detect hateful messages in the embedding vector space. In addition to these lexical features, linguistic and syntactic features can be extracted. These may include the length of messages, average length of words, number of punctuations, Part-of-Speech (PoS) tags and dependency relationships (Nobata et al., 2016).

Recent work on this topic include different neural network architectures to classify text. Here either recurrent neural networks (RNNs) like LSTMs and GRUs, convolutional neural networks (CNNs) or both have been researched (Zhang and Luo, 2018; Zhang et al., 2018; Badjatiya et al., 2017).

3 System Description

Following the latest developments in the field of text classification, our system utilizes different convolutional filters in order to extract features. In the following sections we will describe the steps performed to pre-process the raw text messages and the network architecture used.

3.1 Pre-Processing

Given the raw Twitter messages, several steps of pre-processing are applied. First we follow the suggestions of Pennington et al. (2014), which involves the replacement of certain characters by tags. User mentions are replaced by `"<user>"`, numbers by `"<number>"`, web links by `"<url>"` and repeating characters like `"!!!"` by `"! <repeat>"`. Furthermore, words that are written using uppercase characters only are replaced by the same word in lowercase, followed by `"<allcaps>"`. Hashtags like `"#IllegalImmigrants"` are replaced with `"<hashtag> illegal immigrants"`, splitting the text before each uppercase character. The resulting text is then padded with `"<space>"` to match the length of the longest message.

The sequences are then encoded using pre-trained word embeddings. For English, the 200-dimensional GloVe embeddings from Pennington et al. (2014) are used, because they were trained on Twitter messages and include vectors for the tags mentioned above. For Spanish, the 300-dimensional GloVe embeddings trained on the Spanish Billion Word Corpus (Cardellino, 2016) are used. Unknown words were replaced by `"<unknown>"` in English and `"desconocido"` in Spanish. These embeddings are then directly fed to our neural network classifier.

In order to compare our model to traditional approaches as discussed above, we also implemented a second pre-processing pipeline. This takes the padded sequences, applies stemming and extracts *n*-grams tuples with *n* being in the range from one to four words. These tuples are then vectorized using *TF-IDF*.

3.2 Classification

The detection of hate speech as well as the classification of aggressive behavior and the target harassed can be seen as three independent binary classification tasks. Hence, we can use the same model for each individual subtask and language. The only exception to this is the size of the input, since the the English embedding is 200-dimensional, whereas the Spanish embedding is 300-dimensional.

The network itself employs five different filter types, as shown in Figure 1. All of them are 1-d convolutions, meaning that the windows spread along all dimensions of the embedding size and only move along the words in a message. The first three are traditional convolutional filters with a window size of two, three and four words. These can be seen as n-gram feature extractors. The other two filters have a window size of two, but are dilated with dilation rates of two and three. By skipping words in between two other words, these filters may extract word combinations that may otherwise be missed due to the low importance of the words in the middle. All filter types use a stride of one, same padding and rectified linear units (ReLUs) as activation functions. For each of the five filter type, 100 filters are used. Max-pooling with a filter size of four and a stride of four is performed on all, to reduce dimensionality. In the next layer, the filter maps are concatenated and global max-pooling is performed. This flattens the filters in a non-parametric way and extracts the most pronounced features along all filters. Finally, these features are fully connected

Model	EN	ES
CNN + DIL	**0.78**	**0.82**
CNN	0.77	0.80
SVM	0.63	0.68
LogReg	0.65	0.68
SVM (*n*-gram *TF-IDF*)	0.76	0.81
LogReg (*n*-gram *TF-IDF*)	0.76	0.77

Table 1: Comparison of model performance (F1-score) on the training set for subtask 1 (hate speech detection).

with one output neuron, which uses the sigmoid activation function. The network is trained for ten epochs using the Adam optimization algorithm (Kingma and Ba, 2014) and a batch size of 32.

We found this model to yield the best performance, comparing it to various other approaches. For the test set we selected ten percent of the training data points randomly and stratified. The results of this comparison can be seen in table 1. In order to evaluate the effectiveness of dilated convolutions we removed them from the model, leaving just the three regular convolutional filter types. As a result the F1-scores dropped by 0.01 in English and 0.02 in Spanish. Furthermore, logistic regression and SVM classifiers were trained both on the word embeddings and the *n*-gram based *TF-IDF* features. On the training data our approach performs slightly better than these. One reason for this very marginal improvement over the traditional approaches may be the size of the training set. In order to train a model with a high number of parameters like ours large data sets are majorly beneficial.

4 Results

Here we show the performance of the *Know-Center* system on the challenge's official test sets and compare it with the performances of the other participants. The results are shown in Tables 3 and 2 for the English and Spanish tasks respectively. The provided rankings refer to the average F1-scores over all subtasks, namely hate speech detection, target classification and aggression classification.

As illustrated, the *Know-Center* system achieved F1-scores of 0.45 and 0.72 in identifying hate speech in English and Spanish. For the target classification, we achieved F1-scores of 0.69 and 0.81. In detecting aggressive behavior, our system

#	Team name	AVG	HS	TR	AG
1	MITRE	0.77	0.76	0.82	0.73
2	Saagie	0.76	0.72	0.81	0.76
3	Atalaya	0.76	0.74	0.81	0.73
4	CIC-2	0.76	0.73	0.80	0.74
5	INGEOTEC	0.75	0.71	0.82	0.74
10	**Know-Center**	**0.74**	**0.72**	**0.81**	**0.70**
15	SVC baseline	0.74	0.70	0.78	0.73
26	MFC baseline	0.40	0.37	0.42	0.41

Table 2: F1-scores for each subtask in Spanish (subtask 1: hate speech detection (HS), subtask 2: target (TR) and aggression (AG) classification) as well as the overall average (AVG) per team, including their rank (#)

#	Team name	AVG	HS	TR	AG
1	scmhl5	0.63	0.60	0.71	0.59
2	alonzorz	0.61	0.52	0.75	0.57
3	MITRE	0.61	0.53	0.74	0.58
4	SINAI-DL	0.61	0.52	0.71	0.60
5	YNU_NLP	0.61	0.50	0.71	0.62
17	SVC baseline	0.58	0.45	0.70	0.59
20	**Know-Center**	**0.57**	**0.45**	**0.69**	**0.57**
42	MFC baseline	0.42	0.37	0.45	0.45

Table 3: F1-scores for each subtask in English (subtask 1: hate speech detection (HS), subtask 2: target (TR) and aggression (AG) classification) as well as the overall average (AVG) per team, including their rank (#)

achieved F1-scores of 0.57 and 0.70.

This is particularly interesting when comparing these results with the once obtained on the training data. Here the Spanish model performs similarly, but the English model does not. In general, the F1-scores obtained in the Spanish subtasks are better across all teams, even though less teams participated in it. One reason for that may be differences between training and test set. Besides that, our model uses high dimensional features, given the size of the word embeddings and the message length. For this, the size of the training set is very small, which makes it difficult to train a model with a high number of parameters such as ours. Even though a validation set was used during training, the possible homogeneity of the training set may have led to an over-fitting of the model.

5 Conclusion

We framed the tasks an a binary text classification problem, for which we developed a classification method that we used to participate in both subtasks of task 5 of the SemEval-2019 workshop. The classifier makes use of word embeddings and CNNs to identify hate speech in Twitter messages, determine the target and aggressive behavior. The same pre-processing and network architecture has been used for all tasks and languages. Averaged over all subtasks we achieved F1-scores of 0.57 and 0.74 for English and Spanish respectively.

Acknowledgments

The Know-Center GmbH Graz is funded within the Austrian COMET Program - Competence Centers for Excellent Technologies - under the auspices of the Austrian Federal Ministry of Transport, Innovation and Technology, the Austrian Federal Ministry of Economy, Family and Youth and by the State of Styria. COMET is managed by the Austrian Research Promotion Agency FFG.

References

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Cristian Cardellino. 2016. Spanish Billion Words Corpus and Embeddings.

Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. 2012. Detecting offensive language in social media to protect adolescent online safety. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 71–80. IEEE.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th international conference on world wide web*, pages 29–30. ACM.

Heather Hensman Kettrey and Whitney Nicole Laster. 2014. Staking territory in the world white web an exploration of the roles of overt and color-blind racism in maintaining racial boundaries on a popular web site. *Social Currents*, 1(3):257–274.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ritesh Kumar, Atul Kr Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 1–11.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in discriminating profanity from hate speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30(2):187–202.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153. International World Wide Web Conferences Steering Committee.

Alexandra Olteanu, Carlos Castillo, Jeremy Boy, and Kush R. Varshney. 2018. The effect of extremist violence on hateful speech online. In *Twelfth International AAAI Conference on Web and Social Media*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Sara Owsley Sood, Judd Antin, and Elizabeth Churchill. 2012. Using crowdsourcing to improve profanity detection. In *2012 AAAI Spring Symposium Series*.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval).

Ziqi Zhang and Lei Luo. 2018. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *arXiv preprint arXiv:1803.03662*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting hate speech on twitter using a convolution-gru based deep neural network. In *European Semantic Web Conference*, pages 745–760. Springer.

LT3 at SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter (hatEval)

Nina Bauwelinck, Gilles Jacobs, Véronique Hoste and Els Lefever
LT3, Language and Translation Technology Team
Department of Translation, Interpreting and Communication – Ghent University
Groot-Brittanniëlaan 45, 9000 Ghent, Belgium
`firstname.lastname@ugent.be, gillesm.jacobs@ugent.be`

Abstract

This paper describes our contribution to the SemEval-2019 Task 5 on the detection of hate speech against immigrants and women in Twitter (hatEval). We considered a supervised classification-based approach to detect hate speech in English tweets, which combines a variety of standard lexical and syntactic features with specific features for capturing offensive language. Our experimental results show good classification performance on the training data, but a considerable drop in recall on the held-out test set.

1 Introduction

The exponential growth of social media such as Twitter, Facebook, Youtube and community forums has created a variety of novel ways for all of us to communicate with each other, but this opportunity to freely communicate online has unfortunately also given a forum to people who want to denigrate others because of their race, colour, gender, sexual orientation, religion, etc. While there has been an increasing interest in automatic hate speech detection in social media, the problem is far from solved, partly due to the low consensus on what exactly constitutes hate speech, how it relates to offensive language and bullying and thus the low reliability of hate speech annotations (Ross et al., 2017). Davidson et al. (2017) for example observe that their classifications of hate speech tend to reflect their own subjective biases: while racist and homophobic insults are considered hateful, they tend to see sexist language as merely offensive. When we consider the different approaches that address hate speech, we can observe that -apart from simple methodologies that rely on lookup in a dictionary of hateful terms (Tulkens et al., 2016) - most methods cast the problem as a supervised classification task either using a more standard machine learning approach or deep learning methods (Pitsilis et al., 2018). This was also the approach we took for our hate speech detection system.

We participated for both subtasks proposed for English for Task 5 (Basile et al., 2019), being TASK A, which was defined as a binary classification task where systems have to predict whether a tweet with a given target (women or immigrants) is hateful or not hateful, and TASK B, where systems are asked first to classify hateful tweets as aggressive or not aggressive, and second to identify the target harassed as individual or generic (i.e. single human or group).

2 System Description

We designed a cascaded classification-based approach, where a first classifier categorizes a tweet as being hateful or not, while in a second step the hateful tweets are classified as (a) being aggressive or not, and (b) the target as being individual or generic. For the second step we built separate classifiers for both subtasks (a) and (b).

2.1 Preprocessing

We applied the Twitter-specific tweetokenize (Suttles, 2013)[1] module for tokenization and preprocessing. With this module, we were able to ensure all unicode emojis would be preserved. It also allowed us to add emoticons to the module's lexicon, to avoid splitting them up. We used the module with standard settings: allcaps were maintained; @mentions were replaced by "USERNAME"; urls replaced by "URL". We decided to preserve hashtags and numbers (but replacing phonenumbers and times by "PHONENUMBER" and "TIME", respectively), as well as quotes and stopwords. Finally, we applied a function to tokenize

[1]`https://github.com/jaredks/tweetokenize`

the hashtags, but this proved insufficient, as it did not tokenize camelcased hashtags correctly.

2.2 Featurization

We aimed to develop a rich feature set that focused on lexical information with some syntactic and non-linguistic features included. This featurization pipeline is based on work in cyberbullying detection and analysis (Van Hee et al., 2018). The whole set of features listed below was used to build all three classifiers. We did not apply any feature selection.

Bag-of-words features: We included binary token unigrams, bigrams and trigrams, along with character trigrams and fourgrams. The latter provide robustness to the spelling variation typically found in social media.

Lexicon features: We computed positive and negative opinion word ratio and overall post sentiment using both the MPQA (Wilson et al., 2005) and Hu and Liu's (Hu and Liu, 2004) opinion lexicons. We added positive, negative and neutral emoji counts based on the BOUNCE emoji sentiment lexicon (Kökciyan et al., 2013). We also included the relative frequency of all 64 psychometric categories in the Linguistic Inquiry and Word Count (LIWC) dictionary (Pennebaker et al., 2007). Furthermore, we included diminisher, intensifier, negation, and "allness" lexicons which relate to a negative mindset in the context of suicidality research (Osgood and Walker, 1959; Gottschalk and Gleser, 1960; Shapero, 2011) as well as a proper name gazetteer.

Syntactic features: Two binary features were implemented indicating whether the imperative mood was used in a post and whether person alternation occurred (i.e. combinations of first and second person pronouns).

2.3 Experimental Setup

As mentioned in Section 2, we built three different classifiers to tackle the various subtasks: (1) determine whether a tweet is hateful or not, (2) for tweets classified as hateful, determine whether the target is individual or generic and (3) for tweets classified as hateful, determine whether the tweet is aggressive or not. As the classification algorithm we used LIBSVM (Chang and Lin, 2011) with linear kernel. For each classification task, we performed a grid search to find the optimal cost parameter using 5-fold cross-validation (CV) on the training data. The resulting hyperparameter ($c = 0,03125$) was applied in four different experimental setups: LIBSVM with RBF kernel not normalized, RBF kernel normalized, linear kernel not normalized and linear kernel normalized. These experiments revealed the setup with the linear kernel using normalized data as the best performing system. Table 1 presents the 5-fold CV results for this system on the training set. The experimental results on the training data show good detection results of hate speech (F-score of 71.7% on the positive class), very good results for the classification of the target as being generic or individual (average F-score of 87.5%) and lower classification performance for the classification of aggressive tweets (average F-score of 66.2%).

2.4 Competition Results and Analysis

Table 2 lists the results of our optimized linear kernel system for Task 1, whereas Table 3 shows the results for all three subtasks. As is clear from the results for task 1, our system undergenerates, resulting in a recall of only 6.8%, as opposed to 74.3% on the training data.

Upon closer inspection of the classification errors made by our system for task 1 (hate speech or not), it is possible to distinguish a few broad error categories. In order to reveal possible causes for our low recall, we will focus on hate speech that was not detected by the classifier.

2.4.1 HS Towards Women

Prediction: NOT, Gold: HS. Firstly, we consider the tweets targeted at women. For most of the tweets misclassified by our system as not containing hate speech, it is unclear which features of the text may have caused confusion, as is illustrated by example 1:

(1) That bitch on the spoiled whore list Thought she could play games with Drake but forfeit

Similar examples include at least one offensive term and are clear instances of hate speech. There are, however, some tweets containing some elements which may explain the erroneous classification.

Insults which contain a metaphorical element (such as "meat" in the example below) are hard to classify for our system:

(2) Open your mouth & take the <u>meat</u> like a hoe you bitch ass

	#instances	Recall	Precision	F-score
	TASK 1: hateful or not			
Hateful	4210	74.3	69.2	71.7
Not Hateful	5790	78.7	82.6	80.6
Macro Avg	10000	76.5	75.9	76.2
	TASK 2: individual or generic			
Individual	1560	83.4	82.8	83.1
Generic	2650	89.9	90.3	90.1
Macro Avg	4210	86.7	86.6	86.6
	TASK 3: aggressive or not			
Aggressive	1763	61.1	53.0	56.8
Not Aggresive	2447	69.1	75.7	72.2
Macro Avg	4210	65.1	64.4	64.5

Table 1: Cross-validation results on the training data with the linear kernel with optimised hyperparameter settings.

	Recall	Precision	F-score
	TASK 1: hateful or not		
Hateful	6.8	55.9	12.1
Not Hateful	96.1	58.6	72.8
Macro Avg	51.4	57.3	42.5
Accuracy	58.5		

Table 2: Final results task 1 of best system (optimized linear kernel with normalization).

	Recall	Precision	F-score
	TASK 1: hateful or not		
Macro Avg	51.4	57.3	42.5
Accuracy	58.5		
	TASK 2: individual or generic		
Macro Avg	53.4	60.9	52.9
Accuracy	81.3		
	TASK 3: aggressive or not		
Macro Avg	50.1	90.1	44.7
Accuracy	80.2		

Table 3: Final results of best system (optimized linear kernel with normalization).

Finally, we have noticed that tweets containing self-directed or self-referential insults are often misclassified as not containing hate speech:

(3) @USERNAME 1o million, one cent less, i am a liei8ng son of a bitch and my mom is a whore

Some of the errors can be explained by the ambiguity inherent in the tweet in question. First of all, our system has some difficulty distinguishing between genuine insults and jocular insults between friends. For example, this tweet has been classified as not containing hate speech:

(4) @USERNAME you hoe you need to be checked for aids bitch.

On the other hand, many tweets have been correctly classified as not containing hate speech. These usually contain positive words like ”love” and ”friend”:

(5) Lana i love you bitch. Put that flag back up hoe [USA flag emoji] #lustfoflife

There are also some edge cases, where there is ambiguity as to the intent of the tweeter. It makes sense that our undergenerating system is not preferred in cases such as these:

(6) @USERNAME Bitch RT me one more time & not answer my text [face with steam from nose]

A second error type of ambiguous tweets is that of the tweet containing an intensifier, which is usually ”bitch”, for example:

(7) i’m so salty i tried to go in a store with no shoes on and the ladies like ”you have to have shoes” BITCH IM NOT VCARRYING A DISEASE

A third type of ambiguity occurs when there is some kind of reference to or citation of an existing text containing offensive language. For instance, Tweets sometimes contain references to well-known vines. Some of these are correctly classified by our system (first example, classified as NOT HS), while others are not (second example, misclassified as NOT HS):

(8) I love you...bitch. I ain't never gonna stop loving you...bitch. You're my best friend in the whole wide world and I can't wait to hug you

(9) when ugly hoes try to come for you .... Bitch DISGUSTING

A fourth category of errors caused by ambiguity occurs whenever the tweeter is making a general remark, characterized by the use of plural form (”bitches”, ”niggas”). These tweets usually begin with a phrase indicating the sentiment of the speaker (”I'm tired of”) and can contain an intensifier.

(10) I'm tired of bitches wearin no clothes then got the nerve to say niggas is insecure like bitch I'm tryn help you not look like a hoe

We would like to finish this section with some general remarks on this part of the task, with the focus on HS targeted towards women. After close inspection of the results, we noted the difficulty of annotating such data. First of all, tweets with jocular intent are not always easy to distinguish. We did observe that tweets containing references to birthdays tended not to be hate speech, for example:

(11) @USERNAME happy birthday bitch! Hope you have a fab day you hoe n get on lifts soon [car emoji] URL

Second, the truly ambiguous tweets, those which confuse both man and machine, appear to be those in which there is potential ironic intent. This is mostly the case whenever there is overexaggeration (the reaction to the colleague's question is disproportionate to the topic of tie dye t-shirts):

(12) This bitch at work had the nerve to ask why are all y'all wearing tye die.. because u lil whore it's lit

Third, lots of tweets include references to real-life situations (e.g. someone quoting an insult they have received); or to memes (like the ”Me: X, Also Me: X” meme in the second example):

(13) Fat whore!!! Ugly bitch!!! Me: URL

2.4.2 HS Towards Immigrants

Prediction: NOT, Gold: HS. Once again, our system undergenerates on detecting hate speech targeted at immigrants. First of all, many tweets contained the offensive term ”shithole”. The fact that this is a compound term may explain why our classifier was unable to pick up on ”shit” as offensive:

(14) @JoShawTx a_meluzzi #BanMuslims #BanIslam #SendThemBack to their shitholes!! #CloseTheBorders

Our classifier was not able to pick up on hate speech in tweets containing a metaphorical meaning. For example:

(15) Bloody foreign hornets coming over here killing our bees. #brexitmeansbrexit #sendthemhome #controlourborders

Finally, it is clear that this part of the task requires a classifier which is better able to pick up on derogatory use of the terms ”invasion/invader”, ”illegal alien” and is able to deal with contracted insults like ”rapefugee”:

(16) @ShadilayAlways Illegal invaders that deserve to be deported, as usual. #DeportThemAll #BuildTheWall #KAG #SupportICE

(17) @AhsanNasir5 @enough14 @AZeckenbiss Gtfo off europe. Here is no place for you, you rapefugee.

It turns out our system was only able to correctly classify the tweet as containing hate speech when it also contained other offensive words:

(18) This fuck is a weak cunt and puy his fellow country men in a bad way with the rapefugee invasion. Deport THEM ALL.

3 Conclusion

We applied a supervised classification-based approach to the task of hate speech detection against women and immigrants, incorporating a variety of standard lexical and syntactic information with specific features for capturing offensive language. Our results revealed good classification performance on the training data, but a lower performance on the evaluation set, with a notable drop in recall for all subtasks. A detailed error analysis revealed a number of tendencies, but inherently ambiguous tweets (irony, references to memes and vines, etc.) appeared to be the major cause of classification errors for hate speech towards women. Hate speech against immigrants seems to be characterized by compound terms containing offensive parts (e.g. ”rapefugee”, ”shitholes”).

References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27:1–27:27. ISSN: 2157-6904.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of ICWSM*.

Louis Gottschalk and Goldine Gleser. 1960. An analysis of the verbal content of suicide notes. *British Journal of Medical Psychology*, 33(3):195–204.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Nadin Kökciyan, Arda Çelebi, Arzucan Özgür, and Suzan Üsküdarl. 2013. BOUNCE: Sentiment Classification in Twitter using Rich Feature Sets. In *Second Joint Conference on Lexical and Computational Semantics (*SEM): Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 554–561, Atlanta, Georgia, USA. Association for Computational Linguistics.

Charles Osgood and Evelyn Walker. 1959. Motivation and language behavior: A content analysis of suicide notes. *The Journal of Abnormal and Social Psychology*, 59(1):58.

James Pennebaker, Roger Booth, and Martha Francis. 2007. Liwc2007: Linguistic inquiry and word count. *Austin, Texas: liwc. net*.

Georgios Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Effective hate-speech detection in twitter data using recurrent neural networks. *Applied Intelligence*.

Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Wojatzki. 2017. Measuring the reliability of hate speech annotations: The case of the european refugee crisis.

Jess Jann Shapero. 2011. *The language of suicide notes*. Ph.D. thesis, University of Birmingham.

Jared Suttles. 2013. tweetokenize. *GitHub repository*.

Stephan Tulkens, Lisa Hilte, Elise Lodewyckx, Ben Verhoeven, and Walter Daelemans. 2016. A dictionary-based approach to racism detection in dutch social media. *Proceedings of the First Workshop on Text Analytics for Cybersecurity and Online Safety (TA-COS 2016)*.

Cynthia Van Hee, Gilles Jacobs, Chris Emmery, Bart Desmet, Els Lefever, Ben Verhoeven, Guy De Pauw, Walter Daelemans, and Veronique Hoste. 2018. Automatic detection of cyberbullying in social media text. *PLOS ONE*, 13:22.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics.

—

ltl.uni-due at SemEval-2019 Task 5: Simple but Effective Lexico-Semantic Features for Detecting Hate Speech in Twitter

Huangpan Zhang, Michael Wojatzki, Tobias Horsmann and Torsten Zesch
Language Technology Lab, University of Duisburg-Essen
{huangpan.zhang, michael.wojatzki}@uni-due.de
{tobias.horsmann, torsten.zesch}@uni-due.de

Abstract

In this paper, we present our contribution to SemEval 2019 Task 5 *Multilingual Detection of Hate*, specifically in the Subtask A (English and Spanish). We compare different configurations of shallow and deep learning approaches on the English data and use the system that performs best in both sub-tasks. The resulting SVM-based system with lexico-semantic features (n-grams and embeddings) is ranked 23rd out of 69 on the English data and beats the baseline system. On the Spanish data our system is ranked 25th out of 39.

1 Introduction

Hateful, abusive, or offending statements which target individuals or groups on the basis of characteristics such as gender, nationality, or sexual orientation are called *hate speech* (Basile et al., 2019). Social media is particularly affected by hate speech, as it is known to poison the communication climate, build up negative sentiment towards groups of people, or even lead to real-life consequences (Warner and Hirschberg, 2012; Waseem and Hovy, 2016; Schmidt and Wiegand, 2017; Wojatzki et al., 2018; Benikova et al., 2017; Ross et al., 2017).

In this work, we present our submission to the *SemEval 2019 Task 5: Multilingual Detection of Hate* (Subtask A) for English and Spanish. The objective in Subtask A was to build a system which is able to predict whether given tweets in English or in Spanish are hateful or not hateful towards women or immigrants.

We develop a hate speech detection system by experimenting with a range of classifiers which are either based on engineered features or on neural network architectures. We systematically compare the performance of these different detection systems and for our final submission (for both English and Spanish) we use the model that performs best on the English training data. Our best system is a SVM equipped with n-gram features and fastText (Mikolov et al., 2018) embeddings. Our system obtains the 23rd rank (out of 69) on the English dataset and the 25th rank (out of 39) on the Spanish dataset.

2 System Description

For our submission, we compare a wide range of different neural and non-neural systems in terms of their performance. Our actual submission system is the system that performed best in this evaluation. We will now describe both our neural network approaches and the feature-engineering approaches for detecting whether tweets are hateful towards women or immigrants (Subtask A). We developed and evaluated the approaches for the English dataset and applied the best performing system as-is to the Spanish data. We now first briefly describe the provided data and then discuss the prediction approaches in more detail.

Dataset In Subtask A, the English training set consists of 9,000 tweets and the development set consists of 1,000 tweets. The Spanish training set consists of 5,000 tweets, the development set consists of 500 tweets. In the test data, there are 2,971 English and 1,600 Spanish tweets. For each tweet, the task organizers provided a binary annotation indicating whether a tweet is hateful or not hateful towards a given target (i.e. women or immigrants). An example for the label hateful (towards immigrants) is the tweet:

> *This immigrant should be hung or shot! Period! Animal. https://t.co/wFcGoLCqJ5*

An example for the label not hateful is the following tweet:

Don't mess with these migrant dads #SkimmLife https://t.co/swVmkTlFRz via @theSkimm.

For more details on the dataset and its creation, we refer to the overview paper of the shared task (Basile et al., 2019).

Preprocessing In almost all of our classification approaches, we vectorize the tweets based on word occurrences. Hence, we tokenize the tweets with the twitter specific tokenizer provided by Owoputi et al. (2013). We decided not to remove or normalize social media specific phenomena such as @-mentions, #-hashtags, URLs, and emojis as we hypothesize that these phenomena may provide useful signals for classification. For example, it is conceivable that a reference to the twitter-handle of Donald Trump (*@realDonaldTrump*) may indicate hatred towards immigrants.

2.1 Feature Engineering Approaches

We now report on those approaches that are based on traditional machine learning algorithms and that represent the train and test instances using manually crafted and engineered features. The explored machine learning algorithms are: SVM (LibSVM by Chang and Lin (2011), XGBoost (Chen and Guestrin, 2016), RandomForest (Witten et al., 2016) and Vowpal Wabbit.[1]

We implement the classifiers using the text classification framework DKPro TC (Daxenberger et al., 2014) which includes all of the above-mentioned classifiers. We use the following features to represent the tweets:

N-grams As a baseline feature, we represent the tweets using word and character n-grams. We experiment with n-gram sizes in the range from 1-3 for word n-grams and 2-5 for character n-grams. To reduce the feature space, we only use the n-grams that are most common in the (English and Spanish) training data. We experiment with the frequency cut-off values of 200, 500 and 1,000.

Hateword lists We hypothesize that the presence of specific hate or insult words gives an indication of whether a tweet constitutes hate speech. Hence, we check if the words in the tweets occur in lists of hate or insult words. We use the word lists provided by Wiegand et al. (2018), which contain a basic word list and a extended word list. There are 1,650 words in basic list with binary labels (abusive or not), and 8,478 words in extended list with a numeric weight. We extract abusive words to use in the following features: a) a **boolean hateful** feature if a posting contains any word contained in the basic list, b) a **hatefulness ratio** of total words to hateful words, and c) the sum of the **hatefulness weights** based on the extended list.

[1] https://github.com/VowpalWabbit/vowpal_wabbit

Sentiment We also suspect that the tone in which a tweet is composed can be an indication for hate speech. For instance, we assume that tweets that have a strong positive sentiment are rarely hate speech. To measure the overall sentiment of tweets, we use the tool by Socher et al. (2013) to compute a sentiment score for each tweet. The computed sentiment score uses a five-degree scale from very positive to very negative.

Word embeddings We use pre-trained word embeddings to enhance our tweet representation with a semantic component. For computing semantic features, we first average the 300-dimensional (Spanish or English) word embeddings provided by Mikolov et al. (2018) of all words of a tweet. Next, we use every dimension of the averaged vector as a feature.

2.2 Neural Network Approaches

Besides traditional machine learning approach, we also experiment with neural network architectures: multilayer perceptrons (MLP), convolutional neural networks (CNN), bi-directional LSTMs and a combination of LSTMs and CNNs (LSTM + CNN). We initialize all setups with the 300-dimensional word embeddings provided by Mikolov et al. (2018), which were trained on the common crawl corpus. Furthermore, in all setups, we use a dropout of 0.25 after the embedding layer and update network weights using the *Adam* optimizer (Kingma and Ba, 2014). For all architectures, we have optimized the hyperparameters (e.g. number and size of layers) on an held-out development set. We here report only the best-found parameterization.

MLP Besides the final softmax layer, our MLP has a total of 6 densely connected layers. Starting from the input, the layers have 256, 128, 64, 32, 16 and 8 nodes. We use *relu* as activation function in all layers.

CNN Our CNN uses three stacked convolutional layers that use a filter size of two. The first layer has 128 nodes, the second 64 and the third 32. Subsequently, we apply *max pooling*, a dense layer with ten nodes and the final softmax classification layer.

LSTM At the core of our LSTM is a bidirectional LSTM layer with 128 nodes. This layer is followed by two dense layers (40 and 10 nodes) and the softmax layer.

LSTM + CNN For the combination of LSTM and CNN, we put our CNN model on top of LSTM model.

All of the above-described architectures are implemented using deepTC (Horsmann and Zesch, 2018) with the Keras (Chollet et al., 2015) and Tensorflow (Abadi et al., 2015) backend.

BERT We also experiment with Bidirectional Encoder Representations from Transformers (BERT), which recently excelled in a number of NLP tasks (Devlin et al., 2018). For our experiments, we use the provided pre-trained multilingual-cased BERT-Base model,[2] a maximum sequence-length of 128 and batches of 32 instances. In the described configuration, BERT yields an accuracy of 0.66 after fine-tuning for the second time. As we observe that the performance of BERT begins to decrease from the third fine-tuning, we do not fine-tune the model furthermore.

3 Model Selection and Results

We evaluate each of the proposed prediction approaches in a 10-fold cross-validation on the English training dataset to determine the best performing one. As baseline, we use an SVM equipped with word unigram feature.

For all our approaches, we optimize the hyperparameters (e.g. SVM's slack variable or number of layers in neural networks) and feature configurations (e.g. frequency cut-offs for n-gram features) on the training data and report the best performance for each approach. We start with fine-tuning the n-gram features. We test a wide range of different combinations of n-gram sizes and frequency cut-offs with different classifiers. We report the results in Table 1. We use **wn** and **cn** as the abbreviations of word n-grams and character n-grams.

	macro-F_1	Best n-gram Combination

LibSVM	0.780	wn=1 / topk=1000
		cn=2-4 / topk=200

Random Forest	0.771	wn=1-2 / topk=500
		cn=2-4 / topk=1000

XGBoost	0.764	wn=1-2 / topk=1000
		cn=2-5 / topk=1000

Vowpal Wabbit	0.742	wn=1-3 / topk=1000
		cn=2 / topk=200

Table 1: Results for Fine-tuned n-gram Features.

	macro-F_1	accuracy
Baseline	0.693	0.692
LibSVM	0.787	0.794
LSTM + CNN	0.744	0.768
MLP	0.741	0.750
CNN	0.740	0.746
LSTM	0.674	0.688
BERT	0.660	0.660

Table 2: Results for 10-folds Cross-validation on the Training Dataset for English.

We find that SVM has the overall best performance based on cross-validation, and we continue our experiment (hateword lists, sentiment, word embeddings) using LibSVM with the best n-gram setup. We compare this best feature-engineered system in Table 2 with the neural approaches.

Overall, we observe that the approaches based on feature engineering tend to outperform the neural approaches. As our SVM classifier performs best, we select it as our official submission and also apply it to the Spanish data. Interestingly, in our experiments, BERT and and LSTM perform worst by a considerable margin. However, the combination of LSTM and CNN shows to be competitive with feature engineering approaches.

In Table 3, we show how our system performs on the official test data. We observe a dramatic drop of 30.5 percentage points between performance on the English training and test set. We attribute this loss to the over-fitting to the training data. Nevertheless, our system is able to outperform the most frequent class baseline substantially and especially on the Spanish data the absolute difference to the top-scoring system is low (about 3 percentage points). This means that our system is indeed effective in the task at hand, but also that hate speech detection is a very challeng-

[2] https://storage.googleapis.com/bert_models/2018_11_23/multi_cased_L-12_H-768_A-12.zip

	English	Spanish
Most Frequent Class	0.367	0.370
SVM (baseline)	0.451	0.701
SVM (ours)	0.475	0.696
Top-scoring Team	0.651	0.730

Table 3: Results in Terms of macro-F_1 on the English and Spanish Testset.

Feature Set	macro-F_1
All Features	0.785
-N-grams	0.724
-Word Embeddings	0.778
-Hatefulness Ratio	0.785
-Boolean Hateful	0.785
-Sentiment	0.787
-Hatefulness Weights	0.787

Table 4: Feature Ablation for Our SVM Classifiers.

ing task.

Feature ablation To understand how important the individual features are for our system's performance, we conduct an ablation test for our feature set. We show the results of this ablation in Table 4. The results show that the absence of all features except n-grams and word embeddings leads to an improvement in performance. Consequently, we only use n-grams and word embeddings for our final model. The results also show that n-grams are the most important feature for our model.

4 Distribution of Hate Indicators

When comparing the performance of our system between the training data and test data, we notice a dramatic drop of 30.5 percentage points on macro-F_1. To better understand this drop, we examine the distribution of words for which we suspect that they are good indicators for hate speech – i.e. words which both occur frequently in the data and are commonly seen as a highly offensive words. We examine a frequency distribution of all words and find that the word *'bitch'* meet these criteria. However, the distribution of this word is significantly different in train data and test data. To see whether this is a special case, we examine another high frequency word *'fuck'*. The result is shown in Table 5.

Furthermore, we inspect how these words are distributed across the classes hate speech and not hate speech in both the train and the test set. We visualize this analysis in Table 6.

	Train	Test
Bitch	1,115 in 9,000	1,134 in 2,971
Fuck	675 in 9,000	260 in 2,971

Table 5: Distribution of postings contain *Bitch* and *Fuck*.

	Train		Test	
	Hate Speech		Hate Speech	
Word	Yes	No	Yes	No
Bitch	0.78	0.22	0.44	0.57
Fuck	0.59	0.41	0.57	0.44

Table 6: Hate/not-hate Class Distribution of Postings Contain *Bitch* and *Fuck*.

For the word *'bitch'*, we observe that – in the training data – its occurrence is strongly correlated (the probability is about 0.8) with the class hate speech. In the test set, however, this correlation is considerably weaker. As a result, it is very likely that our classifier will learn that *'bitch'* is a strong evidence for hate speech. As the correlation is different in the test data, this heuristic is likely to lead to misclassification. We conclude that our classifier, which makes strong use of lexical features, is too sensitive to such distributions. Note, that we do not find such a shift for the word *'fuck'*.

5 Conclusion

We present **ltl.uni-due** our submission to SemEval 2019 Task 5 *Multilingual Detection of Hate*. For building our system, We systematically compare a wide range of approaches – including neural network approaches such as LSTMs and BERT and approaches which are based on feature engineering. In our experiments a comparably simple classifier – a SVM equipped with lexico-semantic features (n-grams and word embeddings) – outperforms all other approaches. A comparison between performance on training and test data as well as a quantitative analysis of the dataset shows that our comparably simple classifier is prone to over-fitting, but nevertheless delivers competitive performance in this highly challenging task.

Acknowledgments

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under grant No. GRK 2167, Research Training Group "User-Centred Social Media".

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Darina Benikova, Michael Wojatzki, and Torsten Zesch. 2017. What Does This Imply? Examining the Impact of Implicitness on the Perception of Hate Speech. In *International Conference of the German Society for Computational Linguistics and Language Technology*, pages 171–179. Springer.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: A Library for Support Vector Machines. *Acm Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A Scalable Tree Boosting System. In *Proceedings of the 22Nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM.

François Chollet et al. 2015. Keras. https://keras.io.

Johannes Daxenberger, Oliver Ferschke, Iryna Gurevych, and Torsten Zesch. 2014. DKPro TC: A Java-based Framework for Supervised Learning Experiments on Textual Data. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 61–66. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Tobias Horsmann and Torsten Zesch. 2018. DeepTC – An Extension of DKPro Text Classification for Fostering Reproducibility of Deep Learning Experiments. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 2539 – 2545.

Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint:1412.6980*, pages 1–13. Accessible at https://arxiv.org/abs/1412.6980; last accessed November 27 2018.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in Pre-Training Distributed Word Representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improvedppart-of-Speech Tagging for Online Conversational Text with Word Clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–390.

Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Wojatzki. 2017. Measuring the Reliability of Hate Speech Annotations: The Case of The European Refugee Crisis. *arXiv preprint arXiv:1701.08118*.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

William Warner and Julia Hirschberg. 2012. Detecting Hate Speech on the World Wide Web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26. Association for Computational Linguistics.

Zeerak Waseem and Dirk Hovy. 2016. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93.

Michael Wiegand, Josef Ruppenhofer, Anna Schmidt, and Clayton Greenberg. 2018. Inducing a Lexicon of Abusive Words–a Feature-Based Approach. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1046–1056.

Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. 2016. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.

Michael Wojatzki, Tobias Horsmann, Darina Gold, and Torsten Zesch. 2018. Do Women Perceive Hate Differently: Examining the Relationship Between Hate Speech, Gender, and Agreement Judgments. In *Proceedings of the Conference on Natural Language Processing (KONVENS)*, pages 110–120.

MineriaUNAM at SemEval-2019 Task 5: Detecting Hate Speech in Twitter using Multiple Features in a Combinatorial Framework

Luis Enrique Argota Vega
Posgrado en Ciencia e Ingeniería de la Computación
Universidad Nacional Autónoma de México
Ciudad de México, México
luiso91@comunidad.unam.mx

Jorge Reyes-Magaña
Facultad de Matemáticas
Universidad Autónoma de Yucatán
Mérida, Yucatán, México
Universidad Nacional Autónoma de México
Ciudad de México, México
jorge.reyes@correo.uady.mx

Helena Gómez-Adorno
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas
Universidad Nacional Autónoma de México
Ciudad de México, México
helena.gomez@iimas.unam.mx

Gemma Bel-Enguix
Grupo de Ingeniería Lingüística
Instituto de Ingeniería
Universidad Nacional Autónoma de México
Ciudad de México, México
gbele@iingen.unam.mx

Abstract

This paper presents our approach to the Task 5 of Semeval-2019, which aims at detecting hate speech against immigrants and women in Twitter. The task consists of two subtasks, in Spanish and English: (A) detection of hate speech and (B) classification of hateful tweets as aggressive or not, and identification of the target harassed as individual or group. We used linguistically motivated features and several types of n-grams (words, characters, functional words, punctuation symbols, POS, among others). For task A, we trained a Support Vector Machine using a combinatorial framework, whereas for task B we followed a multi-labeled approach using the Random Forest classifier. Our approach achieved the highest F1-score in sub-task A for the Spanish language.

1 Introduction

Hate speech is defined as any communication that disparages a person or a group based on some characteristics. Given the enormous amount of content generated by users on the web, and in particular in social networks, the problem of detecting hate speech is becoming fundamental. Early detection of this kind of language can help to limit its dissemination over the web and to fight against misogyny and xenophobia.

The goal of the task (Basile et al., 2019) is to detect hate speech on Twitter in a multilingual perspective, for Spanish and English. The task is divided into two related subtasks for each of the languages: (task A) detection of hate speech, and (task B) identifying whether the objective of hatred is a person or group of people. In addition, this second task questions if the author of the message pretends to be aggressive, harmful or even incites violence, in several aspects.

From a machine learning perspective, the task can be seen as a binary classification problem. In order to solve the tasks, we evaluated several machine learning algorithms: Support Vector Machines, Logistic Regression, Multinomial Naive Bayes, Decision Trees and, Random Forest.

For text representation, we extracted linguistically motivated patterns and several types of n-grams (characters, words, syntactic and, aggressive words, among others). The pre-processing steps and the experiments carried out to solve this task are explained in the following sections.

2 Related work

In recent years, the automatic detection of aggressive behavior in social media is gaining a lot of attention. This is consistent with political and social concern about hatred and harassment through these media. Several evaluation campaigns have been recently organized related to hate speech detection such as the hate speech identification task at Evalita (Bosco et al., 2018), the aggressiveness detection task at IberEval (Álvarez-Carmona et al., 2018), and the misogyny identification task (Anzovino et al., 2018) at Evalita (Fersini et al., 2018), among many others.

Our work is based on previous work on aggressive detection of tweets in Mexican Spanish (Gómez-Adorno et al., 2018), which was presented in the MEX-A3T 2018 Workshop (Álvarez-Carmona et al., 2018). It follows a classical machine learning approach, a logistic regression algorithm is trained on linguistically motivated characteristics and various types of n-grams (characters, words, syntactic and aggressive words). Furthermore, an oversampling technique (SMOTE) is used to overcome the problem of unbalanced data, which allowed them to achieve better results in the training corpus, but did not generalize well in the test corpus.

When concerning to hate speech detection related methodologies, Djuric et al. (2015) presented a list of criteria based on the critical race theory to identify racist and sexist slander, whereas Chatzakou et al. (2017) implemented a solid methodology for the extraction of text, user and attributes based on a social media network.

Djuric et al. (2015) used the generated list to annotate a publicly available corpus of more than 16k tweets. They analyzed the impact of various extra-linguistic features along with character n-grams for the detection of hate speech. In turn, they elaborated a dictionary based on the most indicative words in their data.

Chatzakou et al. (2017) studied the properties of bullies and aggressors, and the characteristics that distinguish them from normal users. They found that stalkers post with less frequency, participate in fewer online communities and are less popular than users with standard models of behaviour. Their research shows that machine learning classification algorithms can accurately detect users who exhibit bullying and aggressive behavior, with more than 90% of accuracy.

3 Corpus

For the development phase of the competition, the organizers (Basile et al., 2019) distributed the tweets in four files. Statistics referring the corpus for Spanish and English are presented in Table 1.

For the competition evaluation phase, we have put together the training and test corpus of the competition development phase, in table 2 we present details of the resulting corpus.The test corpus of this phase consists of 1,600 tweets for Spanish and 3,000 for English.

4 Methodology

This section shows in detail how tweets are processed for further classification. It is very important the text to be processed in an appropriate format, so that its manipulation can be done in a simpler and less complex way and an optimal precision can be obtained for the automated methods. Additionally, there are several methods for increasing the characteristics of the system, in order to feed the classifier and have more elements when it comes to analyzing your data.

4.1 Pre-processing

Several researchers show that pre-processing is useful for several natural language processing (NLP) tasks (Montes-y-Gómez, 2001; Justicia de la Torre, 2017), especially when the corpus is made of social network data (Pinto et al., 2012; Gómez-Adorno et al., 2016b). Before the extraction of features, the following pre-processing steps are applied in order to improve the representation of n-grams and to reduce the errors of part-of-speech (POS) labeling:

1. The type of single quotes in the English tweets was standardized. This step was prior to the substitution of abbreviations, thus allowing a correct replacement of the equivalent text.

2. All tweets were standardized to lowercase, which avoids having multiple copies of the same words.

3. The mentions to users (@user) were removed.

4. Url's were removed.

5. Emojis were removed.

6. Following the methodology of Gómez-Adorno et al. (2016a), the abbreviations, contractions and, slangs were replaced by the equivalent text for both Spanish and English. It is important to mention that the vocabulary used in these lexical resources is based on social networks.

		Hateful	Individual target	Aggressive	Total
Spanish	Training	1,838 (41.12%)	1,117 (24.99%)	1,485 (33.22%)	4,469
	Testing	222(44.4%)	137 (27.4 %)	176(35.19%)	500
English	Training	3,783 (42.03%)	1,341 (14.89%)	1,559 (17.32%)	9,000
	Testing	427(42.7%)	219 (21.9 %)	204(20.40%)	1,000

Table 1: Corpus statistics for Task A and B in the development phase

	Hateful	Individual target	Aggressive	Total
Training Spanish	2,060 (41.45%)	1,254 (25.23%)	1,661 (33.42%)	4,969
Training English	4,210 (42.10%)	1,560 (15.60%)	1,763 (17.63%)	10,000

Table 2: Corpus statistics for Task A and B in the evaluation phase

7. Function words (or stopwords) were removed.

8. We replaced the figures that appear in tweet by a single digit (0), since the numbers do not contain semantic information that could be relevant for the task.

9. For hashtags, we had the following criteria: if there was a word detected as hateful/aggressive in the text of the hashtag, the complete hashtag was replaced by that word. If no sign of aggressiveness/hatred was found, then the hashtag was removed.

10. Certain rare and special characters were detected in the tweets and they were replaced by a blank space.

11. In the tweets in English, words such as ”&amp” or the character ”&” were detected, which represented a conjunction, in which case it was replaced by the word ”and”.

12. We deleted punctuation, since it does not add any additional information when processing text data. Therefore, eliminating all cases helps to reduce the size of the training and test data.

13. The sequences of several blank spaces, tabs and line breaks were standardized to a single blank space.

4.2 Features

We took into account several features for the representation of tweets:

- **Character n-grams.** Are capable of detecting the morphological composition of a word (Kulmizev et al., 2017). For natural language processing tasks, where many words are likely to be poorly written, the n-grams of characters are especially powerful (Sanchez-Perez et al., 2017) to detect patterns in such spelling mistakes (Kulmizev et al., 2017). For this approach, a variation of n from 3 to 5 is included.

- **Word n-grams.** Capture the identity of a word and its possible neighbors (Kulmizev et al., 2017). In the experiments, the combination of the n-grams with n varying from 1 to 4 helps to improve the results.

- **POS tags n-grams.** Are sequences of continuous part-of-speech (POS) tags. They capture syntactic information and are useful, for example, to identify the user's intentions in tweets (Gómez-Adorno et al., 2018). We have experimented with various combinations of POS n-grams in the data, finding that a range of 2 to 4 provides the best results in the development set.

- **Aggressive word n-grams.** For this work, we gathered a lexicon of aggressive words containing those words obtained by (Gómez-Adorno et al., 2018) and other words we extracted from the training corpus. We built bigrams and trigrams only with the words of this lexicon.

- **Skipgrams.** We capture groups of 2 words with skips of 2 to 4 words.

- **Function words n-grams.** The frequency of this words is one of the best characteristics to detect hate speech and aggressiveness. Prior to the pre-processing of the corpus, we built function words n-grams from 2 to 4 tokens for both languages. We used the stopwords list from NLTK.

- **N-grams of punctuation symbols.** With this feature we approached the coherence and cohesion to the written text. It helps to detect certain patterns in the analysis of hatred and aggressiveness. Prior to the corpus pre-processing, we built n-grams of 2 to 4 punctuation symbols.

- **Language patterns.** We performed a linguistic analysis of the entire training corpus to detect language patterns that can help to distinguish if the tweets are directed to a person or group of people. We considered two types of patterns: morphological structures, and recurrent lexical patterns.

  - Gómez-Adorno et al. (2018) established that certain morphological combinations can help the classification of tweets. Taking into account this technique, the following combinations were detected: verb + adjective, adjective + verb, noun + adjective, adjective + noun and pronoun + verb.
  - Lexical patterns formed with the series of aggressive words detected in the tweets.

4.3 Classifier

For task A, we used a combinatorial framework (μTC) developed by Tellez et al. (2018). The framework approaches any text classification problem as a combinatorial optimization problem; where there is a search space containing all possible combinations of different text transformations (tokenizers) and weighting schemes with their respective parameters, and, on this search space, a meta-heuristic is used to search for a configuration that produces a highly effective text classifier. Considering all the combinations established in the implementation[1] of (μTC), we added the features described in Section 4.2 and the preprocessing techniques in Section 4.1. Once, the best feature space, we trained an SVM with linear kernel.

For task B, we used the sklearn.multiclass[2] module that implements meta-estimators to solve multi-class and multi-label classification problems, decomposing these problems in binary classification problems. In this sense, the multi-label classification assigns a set of target labels to each sample. In particular, we used the Random Forest algorithm, which showed a better performance than the other algorithms of machine learning that we examined. We performed 10-fold cross-validation experiments to select the best features, weighting scheme and, frequency threshold. The final configuration of the system implements a binary weighting scheme, and considers only those characteristics that occur at least 10 times throughout the corpus and that occur in at least 50 documents in the corpus.

5 Results

The performance measure used for task A is the F1 score. Table 3 and Table 4 show the results using the SVM algorithm for both cases, the development phase and the official results of the final evaluation phase. We obtained the best overal F1 score for task A in Spanish, however, for task A in English the results were much lower.

Position	Team	Dev	Eva
1	Atalaya	-	0.73
1	**mineriaUNAM**	0.80	0.73
3	MITRE	-	0.729

Table 3: Results of task A in Spanish of the development phase (Dev) and the official results of the final evaluation phase (Eva).

Position	Team	Dev	Eva
1	Fermi	-	0.651
2	Panaetius	-	0.571
3	YNU_DYX	-	0.546
...	...	...	...
60	**mineriaUNAM**	0.72	0.384

Table 4: Results of task A in English of the development phase (Dev) and the official results of the final evaluation phase (Eva)

The performance measure used for task B is the Exact Match (EMR). Table 5 and Table 6 show the results using the Random Forest algorithm in the

[1] https://github.com/INGEOTEC/microtc

[2] https://scikit-learn.org/stable/

development phase, as well as the official results of the final evaluation phase. For this sub-task we achieved better results in the English language than in Spanish.

Position	Team	Dev	Eva
1	CIC-2	-	0.705
2	CIC-1	-	0.675
3	MITRE	-	0.671
...	...	...	...
16	**mineriaUNAM**	0.73	0.596

Table 5: Results of task B in Spanish of the development phase (Dev) and the official results of the final evaluation phase (Eva).

Position	Team	Dev	Eva
1	MFC baseline	-	0.58
2	LT3	-	0.57
3	CIC-1	-	0.568
...	...	...	...
11	**mineriaUNAM**	0.54	0.368

Table 6: Results of task B in English of the development phase (Dev) and the official results of the final evaluation phase (Eva).

6 Conclusions

We presented an approach for the detection of hate speech, aggressiveness and harassed objective as individual or group in Twitter in Spanish and English.

We implemented a Support Vector Classification algorithm since it presented a better prediction with respect to other methods. In the case of multi-label classification, the Random Forest algorithm was used. Both algorithms were trained on a combination of linguistic patterns features, a lexicon of aggressive words and different types of n-grams (characters, words, POS tags, aggressive words, word jumps, function words, and punctuation symbols).

The results obtained in Task A when using only the μTC original implementation were improved by the addition of extra features such as the aggressive words n-grams, the punctuation symbols n-grams, and the function words n-grams. Remember that the μTC framework is composed of several easy-to-implement text transformations which allowed us to obtain a high-performance classification model. The main advantage of this technique is that it automatically adjusts the parameters of the model.

Acknowledgments

This paper has been supported by project PAPIIT IA401219 from the Universidad Nacional Autónoma de México.

References

Miguel Á Álvarez-Carmona, Estefanía Guzmán-Falcón, Manuel Montes-y-Gómez, Hugo Jair Escalante, Luis Villaseñor-Pineda, Verónica Reyes-Meza, and Antonio Rico-Sulayes. 2018. Overview of mex-a3t at ibereval 2018: Authorship and aggressiveness analysis in mexican spanish tweets. In *Notebook Papers of 3$^{rd.}$ SEPLN Workshop on Evaluation of Human Language Technologies for Iberian Languages (IBEREVAL), Seville, Spain, September.*

Maria Anzovino, Elisabetta Fersini, and Paolo Rosso. 2018. Automatic identification and classification of misogynistic language on twitter. In *International Conference on Applications of Natural Language to Information Systems*, pages 57–64. Springer.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019).* "Association for Computational Linguistics.

Cristina Bosco, Dell'Orletta Felice, Fabio Poletto, Manuela Sanguinetti, and Tesconi Maurizio. 2018. Overview of the evalita 2018 hate speech detection task. In *EVALITA 2018 6th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian*, volume 2263, pages 1–9.

Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Emiliano De Cristofaro, Gianluca Stringhini, and Athena Vakali. 2017. Mean birds: Detecting aggression and bullying on twitter. In *Proceedings of the 2017 ACM on web science conference*, pages 13–22. ACM.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th international conference on world wide web*, pages 29–30. ACM.

Elisabetta Fersini, Debora Nozza, and Paolo Rosso. 2018. Overview of the evalita 2018 task on automatic misogyny identification (ami). In *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA18), Turin, Italy.*

Helena Gómez-Adorno, Gemma Bel-Enguix, Gerardo Sierra, Octavio Sánchez, and Daniela Quezada. 2018. A machine learning approach for detecting aggressive tweets in spanish. In *In Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018), CEUR WS Proceedings*.

Helena Gómez-Adorno, Ilia Markov, Grigori Sidorov, Juan Pablo Posadas-Durán, and Carolina Fócil Arias. 2016a. Compilación de un lexicón de redes sociales para la identificación de perfiles de autor. *Research in Computing Science*, 115:19–27.

Helena Gómez-Adorno, Ilia Markov, Grigori Sidorov, Juan-Pablo Posadas-Durán, Miguel A Sanchez-Perez, and Liliana Chanona-Hernandez. 2016b. Improving feature representation based on a neural network for author profiling in social media texts. *Computational intelligence and neuroscience*, 2016:2.

Artur Kulmizev, Bo Blankers, Johannes Bjerva, Malvina Nissim, Gertjan van Noord, Barbara Plank, and Martijn Wieling. 2017. The power of character n-grams in native language identification. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 382–389.

Manuel Montes-y-Gómez. 2001. Minería de texto: Un nuevo reto computacional.

David Pinto, Darnes Vilarino, Yuridiana Alemán, Helena Gómez, and Nahun Loya. 2012. The soundex phonetic algorithm revisited for sms-based information retrieval. In *II Spanish Conference on Information Retrieval CERI*.

Miguel A Sanchez-Perez, Ilia Markov, Helena Gómez-Adorno, and Grigori Sidorov. 2017. Comparison of character n-grams and lexical features on author, gender, and language variety identification on the same spanish news corpus. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 145–151. Springer.

Eric S. Tellez, Daniela Moctezuma, Sabino Miranda-Jiménez, and Mario Graff. 2018. An automated text categorization framework based on hyperparameter optimization. *Knowledge-Based Systems*, 149:110–123.

María del Consuelo Justicia de la Torre. 2017. Nuevas técnicas de minería de textos: Aplicaciones.

MITRE at SemEval-2019 Task 5: Transfer Learning for Multilingual Hate Speech Detection

Abigail S. Gertner, John C. Henderson, Amy Marsh,
Elizabeth M. Merkhofer, Ben Wellner and **Guido Zarrella**
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730-1420, USA
{gertner,jhndrsn,amarsh,emerkhofer,wellner,jzarrella}@mitre.org

Abstract

This paper describes MITRE's participation in SemEval-2019 Task 5, *HatEval: Multilingual detection of hate speech against immigrants and women in Twitter*. The techniques explored range from simple bag-of-ngrams classifiers to neural architectures with varied attention mechanisms. We describe several styles of transfer learning from auxiliary tasks, including a novel method for adapting pre-trained BERT models to Twitter data. Logistic regression ties the systems together into an ensemble submitted for evaluation. The resulting system was used to produce predictions for all four HatEval subtasks, achieving the best mean rank of all teams that participated in all four conditions.

1 Introduction

The popularity of social media allows anyone to post their thoughts and opinions for all to see. While the vast majority of these communications are benign, there are those who express hateful or threatening messages online. The identification of hate speech (Fortuna and Nunes, 2018; Schmidt and Wiegand, 2017) on platforms like Twitter is of particular interest for law enforcement and to social media companies who wish to remove accounts with offending content from their sites. Automating the identification of hate speech will allow platforms to flag and remove content much more quickly and effectively.

In this effort we explored neural transfer learning techniques, including word embeddings and fine-tuning of models trained with diverse auxiliary tasks. We built and compared models employing soft attention over sequences and multi-headed self-attention. We also present a novel task to aid in performing additional pre-training of BERT (Devlin et al., 2018) for domain adaptation to Twitter data.

2 Task, Data and Evaluation

HatEval was a shared task organized within SemEval-2019 (Basile et al., 2019). The primary task was detection of hate speech in Twitter, specifically against immigrants and women. This multilingual shared task was organized into two sub-tasks, each presented in both English and Spanish, for a total of four sub-task evaluations.

Task A The first sub-task was simply to identify tweets containing hate speech against immigrants or women. The official metric used for this binary classification task was *macro-averaged F1 score*, in which the F1 scores are calculated for both the positive *hate speech* and negative *not hate speech* classes and then those two scores are averaged.

Task B The second sub-task involved the detection of two specific aspects of hate speech: whether it is targeted at an individual vs. a group of people, and whether it expresses aggression on the part of the author. In this annotation scheme, there is a dependency between these two categories and the hate speech label used in Task A, as tweets could only be labeled as positive for targeting or aggression if they were positive for hate speech. The official metric used for Task B was *Exact Match Ratio* (EMR), which is the proportion of tweets that are labeled correctly for all categories (hate speech, targeting, and aggression). Another way to think of this is as a five-class classification problem where the classes are (H=0, T=0, A=0), (H=1, T=0, A=0), (H=1, T=0, A=1), (H=1, T=1, A=0), (H=1, T=1, A=1). EMR on predicting the three classes separately is equivalent to accuracy on this five-class classification.

Dataset Characteristics The English datasets consisted of 9000 tweets for train, 1000 for dev, and 3000 for test. The Spanish datasets were half the size of the English, with 4500 tweets for train, 500 for dev, and 1500 for test.

Cursory examination revealed drastic differences between the training and test sets, particularly in English. The pejorative term *bitch* appeared in 12% of the training tweets vs. 48% of the test tweets. The hashtags *#BuildThatWall* or *#BuildTheWall* appeared at rates of 6% and 23% in train and test, respectively. Likewise, *#MAGA* was in over 12% of the test set tweets but in under 3% of the training set messages. Thus the English test set appears to be dominated by a handful of heavily represented phenomena.

Different annotation strategies appear to have been used on the training and test sets as well. While tweets mentioning *#BuildThatWall* or *#BuildTheWall* were annotated as hate speech 98% of the time in the training set, this number is 35% on the test set. Similarly, tweets containing *bitch* were labeled as hate speech 78% of the time in the training set vs. 43% of the time in the test set.

The use of hashtags differs markedly between languages. Hashtags are much more frequent in the English training data than the Spanish training data, with English tweets 2.6 times more likely to contain at least one tag, and with tags occurring in English at 4.1 times the rate in Spanish. In the English training data, the most frequent ten hashtags were 23% of the overall total and tended towards American political topics. In Spanish, the top ten tags account for only 8% of the total, exhibiting a much longer and sparser tail.

3 System Overview

For each task, we created an ensemble of systems, each of which independently predicted the classes. The component systems are described in the following eight sections, after which we describe the procedure for building and testing the ensembles. All component systems described below treated Task B as a five-class prediction problem, and with the exception of two BERT-based systems, were trained to address Task A and Task B simultaneously.

Data and resources SemEval organizers provided training and development sets for English and Spanish. Planning to build ensembles, we shuffled and split out 10% of the training for calibrating models in the ensembles (*calibration set* from here on). Components were trained using the remaining 90% of the training sets provided, with hyperparameter search and validation using the full development sets or via cross-validation. We did not use any additional supervised datasets.

The *BiLSTM*, *Name Embedding*, and *Hashtag Prediction* models incorporated pre-trained `word2vec` (Mikolov et al., 2013) language-specific embeddings that we trained on 1558 billion English and 444 million Spanish tweets collected from 2011 to 2018. In both cases we applied `word2phrase` twice to identify phrases of up to four words, and trained a skip-gram model of size 256, using a context window of 10 words and 15 negative samples per example.

For Task A, all of our component systems and ensembles included a post-processing step to select the best threshold score for classifying hate speech in order to achieve the maximum macro-averaged F1 score on the development set.

3.1 BiLSTM with Attention

We trained several heavily regularized single-layer Bidirectional LSTM (Hochreiter and Schmidhuber, 1997) models to learn a tweet representation with soft attention (Bahdanau et al., 2014) over a sequence of pre-trained token embeddings. Hyperparameter experimentation with Spearmint (Snoek et al., 2012) suggested that a shallow network with attention outperformed deeper, stacked networks and networks without attention. Our attention layer learns to weight context-aware representations of each timestep of the input.

We trained one architecture for the English tasks and two architectures for Spanish, although the second was ablated from our Task A ensemble. The models were identical in structure and differed only in hyperparameters. All models were constructed with spatial dropout over a frozen embedding layer, followed by an embedding transform, one bi-directional LSTM layer with dropout, an attention layer, and a fully-connected hidden layer with dropout.

In each of these models, the NLP representation was used as input to a small prediction network of latent predictions and residual connections described in Section 3.5.

3.2 Name embeddings

This model added a name embedding input to our BiLSTM described above, in an effort to better model the demographics of the individuals addressed within a tweet.

We trained our name vectors using the word2vec objective. Each context was made up of multiple usernames a single Twitter user had employed during a multi-year longitudinal sample of random tweets streamed from the platform. This resulted in a vocabulary of approximately two million name pieces, which includes common names as well as alternate spellings using special characters, symbols, emoji, and other text entered in the *user name* field.

We extracted all substrings of at least length 3 from each username mention in a tweet and included any of them that were in our name embedding vocabulary as input to our model. We applied a learned transformation to each embedding and created a weighted combination with an attention layer. This was concatenated with a hidden representation constructed with the BiLSTM architecture described in Section 3.1. This concatenation was the input to the prediction network described in Section 3.5.

The Spanish name embedding was comprised of dropout over frozen embeddings, a dense embedding transform, and an attention layer. For English, only an attention layer over the frozen embeddings was used. The hyperparameters from our best English model were used in the BiLSTM architecture for both languages.

3.3 DeepMoji

The DeepMoji model developed by Felbo et al. (2017) predicts the emoji removed from an English-language tweet text. The authors train their RNN model on 1274 million tweets for a set of 64 emojis. Using varying degrees of fine-tuning and newly initialized layers, they test their distantly supervised models on several benchmark datasets for detecting emotion, sentiment, and sarcasm. The model's best results used their *chain-thaw* fine-tuning method, which iteratively unfreezes and trains layers for the new objective. The authors distribute their trained model for the emoji prediction task.

We experimented with both *chain-thaw* training and models that were frozen until the final layer of abstraction in DeepMoji. The pre-trained model has a vocabulary that omits many of the hashtags and usernames that were important for our task. Our best model used 0.75 dropout over the output of a frozen DeepMoji model and three fully connected layers of sizes 512, 256, and 128 before the annotation constraint adapter. *Chain thaw* models performed poorly and were ablated from our Task A submission. DeepMoji models are only included in our English ensembles.

3.4 Hashtag prediction network

Following Zarrella and Marsh (2016), we implemented a recurrent neural network classifier that was pre-trained via an auxiliary masked hashtag prediction task. We extracted 30 of the top hashtags found in the training data, with 15 selected from both the hate speech positive and negative classes. Then we searched for the fifteen nearest neighbors of each tag via cosine similarity in embedding space, using vectors described in Section 3. After removing duplicates, this resulted in 136 English and 132 Spanish hashtags. We downloaded up to 1,000 recent tweets containing each hashtag from Twitter's public search API, resulting in 11,539 English tweets and 12,504 Spanish tweets. Tweets were stripped of the target hashtag(s), and each corpus was divided into a training and development set using a 90/10 split.

The sequence of vector representations of the tokens in each tweet served as the input to a neural network with a 128 LSTM units followed by a dense softmax layer over the possible candidate hashtags. Both the word embeddings and the recurrent layer were tuned. These models correctly predicted development set hashtags with 50.3% accuracy on the English data and 56.6% accuracy on the Spanish data.

The trained weights were extracted from this network and used to initialize the five-way hate speech classifier for Task B, described in Section 2, which additionally saw as input the one-hot representations of the 600 most frequent unigrams and 300 most frequent bigrams in the training data, each followed by a fully-connected dense layer. The size of each fully connected layer and amount of dropout were experimentally determined using Spearmint (Snoek et al., 2012) to maximize performance on the competition metrics on our development set.

3.5 Annotation constraint adapter

Both Task A and Task B had annotation constraints based on latent variables. In Task A, hate speech (H) was not marked as true unless the tweet was directed at women (W) or immigrants (I). In Task B, aggression (A) and individual targeting (T) were not marked as true unless hate speech directed at women or immigrants was present. Even though W and I are not directly represented in our

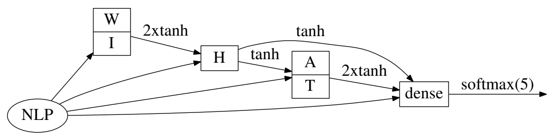


Figure 1: An annotation constraint adapter.

datasets, we believe they are latent variables that can be discovered in the NLP representation. Figure 1 shows an adapter we placed at the end of several systems to encourage the network to learn these constraints. While it doesn't enforce the constraints, it sets up a principled graphical model that encourages the network to learn them. Of course, nothing prevents the network from learning to model other things with this topology. Fair comparisons to stacked dense layers with the same number of parameters showed that the network with this topology performed better.

The upside to the design of a network like this is that the removal of the H switch might yield more general-purpose A and T classifiers.

3.6 Pre-training BERT with Twitter data

Pre-trained language models such as BERT (Devlin et al., 2018) have been demonstrated to achieve state of the art performance on a range of language understanding tasks. BERT uses a transformer encoder model (Vaswani et al., 2017) and pre-trains the model using two complementary objectives: masked language model, and next sentence prediction. The pre-trained model may then be fine-tuned on labeled data (in this case the HatEval dataset) to perform a downstream task.

For English, we used the BERT-Large model, which has 24 layers, 1024 hidden layer size, and 16 self-attention heads. For Spanish, we used the smaller multilingual BERT, with 12 layers, 768 hidden layer size, and 12 self-attention heads. The English BERT is trained on Wikipedia and BooksCorpus (Zhu et al., 2015), while the multilingual model is trained on Wikipedia from multiple languages. As the language in these sources is likely to be quite different from the language commonly used on Twitter, we elected to perform additional pre-training using a corpus of tweets collected during the same time period as the HatEval training dataset (October 2017 - September 2018). All of the pre-training experiments described below started from the TensorFlow model checkpoints downloaded from (Google Research, 2018).

Since the tweets in our collection are not sequential, they cannot be used for the next sentence prediction that BERT uses to learn sentence relationships. We therefore began by running 20k steps of additional pre-training using only the masked language model task.

		none	MLM	descriptions	names
En	A	79.1	**81.2**	79.7	NA
En	B	66.4	**69.0**	67.9	NA
Es	A	80.7	81.9	**83.3**	82.7
Es	B	74.6	75.0	**76.2**	74.4

Table 1: Scores achieved with pre-training schemes. Due to time constraints, the name-based training was only done on Spanish models.

Next, we hypothesized that replacing the next-sentence prediction task with a task involving predicting some attribute of the *author* of the tweet would provide the model with latent information about the nature of tweets that would allow it to discriminate between different classes of tweets more accurately. We performed 20k additional pre-training steps with the *user description* from the author's Twitter profile standing in for the second sequence in the sentence prediction task. In other words, we trained the network to determine whether a given pair of (tweet text, author description text) were sampled from the same tweet. Finally, we pre-trained a BERT model with the *screen name* of the Twitter user as the secondary prediction task.

Table 1 shows the validation scores for our five-class model under our different pre-training schemes: No additional, pre-training on masked LM only, pre-training MLM + Twitter user descriptions, pre-training MLM + Twitter user screen names. Additional pre-training resulted in increased validation scores on all four tasks, and incorporating user descriptions in place of the next sentence prediction task further resulted in increased scores for both Spanish tasks.

3.7 Maximizing ensemble diversity

During development, we noticed some of the neural network models with high capacity had significantly variance in prediction accuracy based on training with different subsets of the training data, hyperparameter settings or just differences in parameter initialization. Such variance would suggest using model bagging (Breiman, 1996) or other form of variance reduction. However, given the relatively long training times for some of the neural network models, especially those based on

$\sigma\|w\|$	w	feature
0.81	-2.57	*bitch*
0.30	-1.52	*whore*
0.29	-1.12	_bitch_
0.27	-0.97	*women*
0.26	0.53	*URL*
0.23	-1.37	*hoe*
0.23	-0.67	*!*
0.19	-0.91	*her*
0.19	0.72	*immigrant*
0.17	-0.88	*#buildthatwall*
0.17	0.34	//t.co/
0.15	0.74	[*URL*,*URL*]
0.15	-0.66	#BuildT
0.14	-0.77	*she*
0.14	-0.30	*a*
0.13	-0.61	*woman*
0.13	0.36	*i*
0.12	-0.63	Illegal
0.12	-0.62	*immigrants*
0.12	-0.40	*this*
0.12	-0.39	igrants
0.10	0.63	[*not*,*all*]
0.10	0.63	[*all*,*men*]

Table 2: Top LR *word* and character features.

BERT, using ensemble methods such as bagging directly proved too cumbersome as part of the model development workflow. Instead, we employed a form of negative correlation learning (Liu and Yao, 1999) to train a small ensemble of neural network classifiers within a single architecture. A term was added to the fine tuning cross entropy loss function which encouraged diversity among all pairs of classifiers following Opitz et al. (2016).

3.8 Logistic Regression

Logistic regression (LR) systems were developed as a baseline against which the neural approach would be compared. Had annotators used very simple features such as words or phrases to make decisions, they would have been found in the course of LR training. Some of the systems were good enough to include in the final ensembles.

The vocabulary of the LR system was limited to the training set. Many feature sets were explored during model search. The best models preferred feature *sets* rather than *counts* or *term frequencies*. Word n-grams of length 1-3 and character n-grams to length 8 were all considered, along with skip bigrams. The specifics of the best resulting feature sets are in Table 3. Table 2 shows the most important features from an English Task A LR system, sorted by feature *influence*, the product of feature function standard deviation and model weight. The second column is model weight, with negative weights contributing to a (H=1) decision.

In all cases, a bias term was added and Liblinear (Fan et al., 2008) was used to compute the model. L2 regularization was used to encourage generalization. Cross-validation was used to pick the regularization parameters.

3.9 Ensemble

Many systems were created, and final ensembles were constructed by incremental ablations. An initial *all-in* ensemble was created and tested, then it was tested with each component removed. This process was iterated on the best performing ablated sets until gains were no longer observed. Approximately two thousand total ensembles were created through the ablative search. Two systems were ablated in Task A EN, three in Task A ES, one in both Task B conditions. Those systems are not described in this paper.

Ensembles were constructed using logistic regression on either the classifier outputs or the classifier outputs and final probabilities from the model. One oddity to note is that the ensembles using the probabilities performed better for Task A and the ensembles ignoring the probabilities performed better in Task B.

Table 3 shows ensemble compositions for each of the four tested conditions. The first column, labeled *influence*, indicates the influence that the particular component has on the ensemble. It is the number of cases in which that component's contribution *changes* the outcome of the ensemble. It is calculated by zeroing out all LR weights for that particular component and noting the difference. In English, the BERT models had the most influence, while in Spanish, the influence was more evenly distributed across the components.

4 Results

Table 3 shows performance of our component models and ensembles. The *calibration set factored* column shows the performance of the component on our calibration data. This is the *macro averaged F1* score for Task A and *Exact Match Ratio* for Task B. The *calibration set ablated* column shows the performance of the ensemble when that component is removed and the ensemble parameters are re-optimized. Finally there are the scores we calculated after the evaluation period for each of our components using the released reference sets.

The official scores achieved by our ensembles

language	task	calibration set influence	calibration set factored	calibration set ablated	test set	component
En	A		86.5		49.6	combo
		369	**84.1**	**84.6**	**58.5**	BERT w/ MLM, 5-class, constraint adapter
		65	78.6	85.7	42.1	BiLSTM+Attn
		59	74.5	85.7	48.0	DeepMoji
		51	82.6	86.2	52.9	BERT w/ descriptions, 1-class
		21	81.3	86.3	47.0	BERT ensemble diversity
		19	78.2	86.1	34.2	BiLSTM with name embeddings
		8	76.4	85.9	48.0	LR, ngrams 1-3, len 7 chargrams, lowercase
		6	75.5	85.8	44.2	Hashtag prediction
		5	77.2	85.9	47.6	LR, ngrams 1-3, len 7 chargrams, lowercase
En	B		77.3		39.9	combo
		435	**74.1**	**75.1**	41.0	BERT w/ MLM
		215	71.7	75.9	37.4	BERT w/ descriptions, constraint adapter
		65	70.6	75.8	33.3	BiLSTM+Attn
		55	67.9	76.6	**43.1**	DeepMoji chain-thaw
		41	58.7	76.0	23.2	Hashtag prediction
		35	69.3	76.4	29.2	BiLSTM with name embeddings
		23	65.6	77.0	38.7	DeepMoji
		23	68.6	76.9	41.0	LR, ngrams 1-2, len 7 chargrams, lowercase
		16	68.2	76.7	41.1	LR, unigrams, len 5 chargrams, lc, skip bigrams
Es	A		87.3		72.9	combo
		90	81.1	**84.9**	**74.3**	BERT w/ names, 5-class
		79	77.9	85.1	73.4	Hashtag prediction
		50	82.1	86.6	73.4	BERT w/ names, 1-class
		45	83.4	85.0	72.0	BiLSTM+Attn
		39	**84.8**	85.3	**74.3**	BERT ensemble diversity
		35	80.7	86.8	73.3	BERT w/ names, 5-class, constraint adapter
		32	79.6	85.0	71.7	LR, ngrams 2-3, len 4 chargrams, lc
		17	82.2	85.0	73.4	BiLSTM with name embeddings
		15	81.4	86.0	73.7	BERT w/ descriptions, 1-class
Es	B		84.7		67.1	combo
		48	78.4	81.8	59.7	BERT ensemble diversity
		42	77.3	80.7	65.3	BERT w/ descriptions
		40	77.8	82.4	63.6	BiLSTM+Attn
		26	75.8	**80.2**	66.8	BERT w/ names, constraint adapter
		21	75.6	83.6	65.6	BERT w/ names
		20	70.0	82.2	59.4	Hashtag predictions
		20	78.4	82.9	67.4	BiLSTM+Attn
		19	76.4	82.4	**68.8**	LR, ngrams 1-3, chargrams 4-7, lc
		15	76.2	82.0	66.7	BERT w/ descriptions, constraint adapter
		11	**79.1**	83.3	65.6	BiLSTM with name embeddings

Table 3: Ensembles and Components

are 49.6% and 72.9% Macro F1 on HatEval Task A English and Spanish, respectively, and 39.9% and 67.1% EMR on Task B English and Spanish. A full reporting of results is present in Basile et al. (2019). A breakdown of test results shows that our system achieves hate speech detection F1 of 63.9 and 72.7 in English and Spanish, respectively, which ranked 2nd (of 68) and 1st (of 39) within Task A. The rankings within Task B were similar, with mean macro F1 of 61.4 and 77.2 in English and Spanish, respectively, ranking 2nd (of 42) and 1st (of 24). Finally, we note that only 22 out of the 74 participants submitted entries in all four subtasks. Of those 22 teams, these results represent the top mean rank across all subtasks.

5 Conclusion

An ensemble of models was used to classify tweets according to whether they contained hate speech, aggression, and targeting of individuals. The novel contributions include using *name embeddings*, substituting twitter author profile prediction for next sentence prediction in BERT pre-training, and augmenting BERT's fine-tuning loss function with a diversity term to create an ensemble.

There is a discrepancy between the official test set results and our held-out calibration set, particularly in the English subtasks, which we attribute to dataset divergences like those called out in Section 2.

Approved for Public Release; Distribution Unlimited. Public Release Case Number 19-0700

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations Workshop*.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Leo Breiman. 1996. Bagging predictors. *Machine learning*, 24(2).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Comput. Surv.*, 51(4):85:1–85:30.

Google Research. 2018. `https://github.com/google-research/bert`.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yong Liu and Xin Yao. 1999. Ensemble learning via negative correlation. *Neural networks*, 12(10):1399–1404.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*.

Michael Opitz, Horst Possegger, and Horst Bischof. 2016. Efficient model averaging for deep neural networks. In *Asian Conference on Computer Vision*.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

Guido Zarrella and Amy Marsh. 2016. MITRE at SemEval-2016 task 6: Transfer learning for stance detection. In *SemEval@NAACL-HLT*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *2015 IEEE International Conference on Computer Vision (ICCV)*.

OscarGaribo at SemEval-2019 Task 5: Frequency Analysis Interpolation for Hate in Speech Detection

Òscar Garibo i Orts
Universitat Politècnica de València / 46025 València Spain
osgaor@alumni.upv.es

Abstract

This document describes a text change of representation approach to the task of Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter, as part of SemEval-2019[1]. The task is divided in two sub-tasks. Sub-task A consists in classifying tweets as being hateful or not hateful, whereas sub-task B requires fine tuning the classification by classifying the hateful tweets as being directed to single individuals or generic, if the tweet is aggressive or not. Our approach consists of a change of the space of representation of text into statistical descriptors which characterize the text. In addition, dimensional reduction is performed to 6 characteristics per class in order to make the method suitable for a Big Data environment. Frequency Analysis Interpolation (FAI) is the approach we use to achieve rank 5th in Spanish language and 9th in English language in sub-task B in both cases.

1 Introduction

Social media has become a new standard of communications in the last years. Every year more and more people actively participate in the content creation, sometimes under the shield of anonymity. Social media has become a complex communication channel in which usually offensive contents are written. Supervising the content and banning offensive messages currently is a subject of high interest for social media administrators. Offensive speech can be addressed to individuals or groups due to the race, sexuality, religion and some other characteristics. In this task two of these characteristics will be used as target for offensive speech, women and immigrants. This problem will be considered as an Author Profiling task, since the main goal is building a system which would ideally detect author whose content is offensive to women and/or immigrant. Author Profiling is widely studied and some new ideas arise from time to time (Rangel et al., 2016). We have developed a new representation method for text that reduces the dimensionality of the information for each author to 6 characteristics per class. This representation, Frequency Analysis Interpolation, is used to codify the texts for each user and this codified information is used as input data to support vector machines with linear kernel. In a Big Data environment, reducing the number of characteristics from thousands to 6 per class allows an efficient way to deal with high volumes at high speed. With this will in mind a previous method was tested which can be checked at (Garibo, 2018).

[1] alt.qcri.org/semeval2019/

2 Corpus

Two corpora have been created to be used in SemEval Task5, HatEval (Basile et al., 2019). One in each of the 2 different languages which are subject of study (i.e. English and Spanish). For each language, a training and an evaluation datasets have been provided. The contents of both datasets are individual tweets, that have been collected and manually annotated.
The goal of this task is to identify tweets which contain hate against women and immigrants. The task has two related subtasks:

1. Task A. Hate Speech Detection against Immigrants and Women: a two class classification where systems have to predict whether a tweet with a given target (women or immigrants) is hateful or not hateful. This is labeled as a 1 in HS column.

2. Task B. Agressive Behaviour and Target Classification: where systems are asked first

Language	Training	Evaluation
English	10,000	3,000
Spanish	5,000	1,600

Table 1: Number of tweets per dataset.

Language	Training	Evaluation
English	4,210	1,260
Spanish	2,790	660

Table 2: Number of Hate tweets per language.

to classify hateful tweets (e.g., tweets where Hate Speech against women or immigrants has been identified) as agresive or not agresive, labeled as AG column in the datasets, and second to identify the target harassed as individual or generic (i.e. single human or group), labeled as TR column in the datasets.

3 Methodology

Our goal was to develop a method that was language independent and that required no prior knowledge of the language used by the authors. We started implementing Term Frequency (TF) representation for each tweet in the corpus, counting how many times each word appears in each author, each tweet in this case, and globally for all tweets. We denote TF$_{\text{a}}$ as the term frequency vector for author a.

$$TF_{\text{a}} = TF_{(w_1,a)}, TF_{(w_2,a)}, \ldots, TF_{(w_m,a)} \quad (1)$$

TF is used since this way we could represent a priori class dependent probability for each term for each class simply by counting the number of times a term occurs for each class, and dividing this amount by the number of times this term shows for all classes. Let F be the frequency term vector for all classes.

$$F = \sum_{a \in A} TF_{\text{a}} \quad (2)$$

In order to achieve that, one vector per class is generated. The vector length is the number of words in the vocabulary. For each word, we divide the number of times this word shows for this class, and divide it by the number of times the word shows in all classes. We denote C$_{\text{k}}$ as the term frequency vector for class k that belong to the set of all classes K.

HS	AG	TR	Label
0	0	0	000
1	0	0	100
1	0	1	101
1	1	0	110
1	1	1	111

Table 3: Labels for the SVM.

$$C_{\text{k}} = \sum_{a \in A_{\text{k}}} TF_{\text{a}} \forall k \in K \quad (3)$$

These vectors are then used to codify the texts. Each word in the text is substituted by the a priori probability for each class in as many arrays as classes.
Once we have codified the text, six statistic values are calculated for each of the classes:

1. Mean.
2. Standard Deviation.
3. Skewness.
4. First Tertile's length.
5. Second Tertile's length.
6. Third Tertile's length.

At this point, for every author, 6 characteristics per class are calculated and concatenated in a single vector. This vector is used to feed the Support Vector Machines with Linear kernel. LinearSVC support vector machine from Pythons Sklearn library is used to train the model and, of course, to predict the results. In order to provide with the labels for the support vector machines to learn the different labels were concatenated to build a 5 class classifier. In Table 4 the 5 classes which were provided to the support vector machine are shown. The same encoding procedure has to be performed for the test dataset. One vector is created for each author. This vector contains the six characteristic mentioned above for every class, concatenated.

4 Evaluation results

Task A is evaluated using standard evaluation metrics, including accuracy (Acc), precision (P), recall (R) and F1-score (F1), while submissions were ranked by F1-score. The metrics were computed as follows:

$$Acc = \frac{numberofcorrectlypredictedinstances}{totalnumberofinstances} \quad (4)$$

$$P = \frac{numberofcorrectlypredictedinstances}{numberofpredictedlabels} \quad (5)$$

$$R = \frac{numberofcorrectlypredictedinstances}{numberoflabelsinthegoldstandard} \quad (6)$$

$$F1 = \frac{2 * P * R}{P + R} \quad (7)$$

FAI has not achieved great results for Task A. Since the change of representation depends on the vocabulary that is used, subtle sentences which can denote hate in the speech but which are not using explicit offensive vocabulary might have been mislabeled. For example, polysemic words can be causing mislabelling, since FAI only considers the per class term frequency, but no context is taken into account. Because the method is language independent, the differences of performance between both languages (English and Spanish) depends on the term frequency for each class observed for the train dataset.

Task B is evaluated using the Exact Match measure where all the dimensions to be predicted will be jointly considered computing the Exact Match Ratio . Given the multi-label dataset consisting of n multi-label samples (xi,Yi), where xi denotes the i-th instance and Yi represents the corresponding set of labels to be predicted (HS 0,1, TR 0,1 and AG 0,1), the Exact Match Ratio (EMR) will be computed as follows:

$$EMR = \frac{1}{n}\sum_{i=1}^{n} I(Yi, Zi) \quad (8)$$

Where Zi denotes the set of labels predicted for the i-th instance and I is the indicator function.

Our method has performed better in TASK B than Task A. Once we provide with more refined labeling, the method tends to catch better the use of aggressive language. This can be seen in the results for both languages for Task B in tables 5 (English) and 6 (Spanish).

As in Task A, the difference of performance of FAI for Spanish and English datasets depends on the term frequency for all classes. Different results have to be expected for different languages.

Ranking	Participant	EM
	MFC Baseline	0.58
1	ninab	0.57
2	iqaameer133	0.568
3	scmhl5	0.483
4	garain	0.482
5	gertner	0.399
6	amontejo	0.384
7	alonzorz	0.382
8	saagie	0.374
9	OscarGaribo	0.373
⋮	⋮	⋮
	SVC Baseline	0.308
⋮	⋮	⋮
42	abaruah	0.159

Table 4: Task B classification for English language.

Ranking	Participant	EM
1	hammad.fahim57	0.705
2	iqaameer133	0.675
3	gertner	0.671
4	francolq2	0.657
5	OscarGaribo	0.6449
6	kwinter	0.638
⋮	⋮	⋮
12	choal	0.616
	SVC Baseline	0.605
⋮	⋮	⋮
16	Taha	0.593
	MFC Baseline	0.588
⋮	⋮	⋮
24	guzimanis	0.428

Table 5: Task B classification for Spanish language.

5 Conclusions and future work

We have used FAI, a method developed under the scope of Author Profiling tasks to approach HatEval Task. FAI has shown to get better results for multi-class classification in the context of this task. Prior testing performed with our method has been done under different environment, since there were always lots of tweets (minimum 100) per author. Thus, there was much more vocabulary to learn from, and more vocabulary per author. We have to point that our method can easily be updated with new data, since the only required task to be done is recomputing the a priori probability vectors once the new labeled data is available, and train the machine learning algorithm, support vector machines in this specific case. As future work we think of exploring new configurations of our method. Since only the last submission was evaluated we still do not know if we can go any further and do better with simple adjustments. One of the immediate ones is to remove some of the vocabulary from the vocabulary we use to codify the tweets. We have seen in our in house testing that some problems require the more the better vocabulary, for example age identification, whereas some others work better if low used words are removed from the vocabulary, for example removing words used by less than 1% of the authors.

References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Òscar Garibo. 2018. A big data approach to gender classification in twitter. In *CLEF 2018 Labs and Workshops. Notebook Papers. CEUR Workshop Proceedings. CEUR-WS.org/Vol-2125/paper*$_2$04*.pdf*.

Francisco Rangel, Marc Franco-Salvador, and Paolo Rosso. 2016. A low dimensionality representation for language variety representation. In *Linguistics and Intelligent Text Processing, CICLing-2016, Springer-Verlag, Revised Selected Papers, Part II, LNCS(9624)*, pages 156–169.

STUFIIT at SemEval-2019 Task 5: Multilingual Hate Speech Detection on Twitter with MUSE and ELMo Embeddings

Michal Bojkovský
Faculty of Informatics and Information
Technologies STU in Bratislava
Ilkovičova 2, Bratislava
m.bojkovsky@protonmail.com

Matúš Pikuliak
Faculty of Informatics and Information
Technologies STU in Bratislava
Ilkovičova 2, Bratislava
matus.pikuliak@stuba.sk

Abstract

We evaluate the viability of multilingual learning for the task of hate speech detection. We also experiment with adversarial learning as a means of creating a multilingual model. Ultimately our multilingual models have had worse results than their monolignual counterparts. We find that the choice of word representations (word embeddings) is very crucial for deep learning as a simple switch between MUSE and ELMo embeddings has shown a 3-4% increase in accuracy. This also shows the importance of context when dealing with online content.

1 Introduction

The Internet has been surging in popularity as well as general availability. This has considerably increased the amount of user generated content present online. This has, however, brought up a few issues. One of the issues is hate speech detection, as manual detection has been made nearly impossible by the quantity of data. The only real solution is automated hate speech detection. Our task is detection of hate speech towards immigrants and women on Twitter (Task A).

Hate speech can be defined as "Any communication that disparages a person or a group on the basis of some characteristic such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics." (Basile et al., 2019) This proves to be a very broad definition, because utterances can be offensive, yet not hateful (Davidson et al., 2017). Even manual labeling of hate speech related data is notoriously difficult as hate speech is very subjective in nature (Nobata et al., 2016; Waseem, 2016).

The provided dataset consists of collected messages from Twitter in English or Spanish language. Hate speech datasets are very prone to class imbalances (Schmidt and Wiegand, 2017). The provided dataset does not suffer from this problem. The English data contains 10,000 messages with 42.1% of the messages labeled as hate speech. The Spanish data contains 4969 messages and similarly to the English part, 41.5% were labeled as hate speech. This gives us a dataset with 14969 messages of which 6270 are categorized as hate-speech. We have not used any additional sources of training data for our models. More information about the data can be found in the Task definition (Basile et al., 2019).

Most research dealing with hate speech has been done in English due to labelled dataset availability. However, this issue is not unique to English-based content. In our work, we explore multilingual approaches, as we recognize data imbalance between languages as one of major challenges of NLP. Multilingual approaches could help remedy this problem, as one could transfer knowledge from a data-rich language (English) to a data-poor language (Spanish).

1.1 Background

We focus on neural network approaches, as they have been achieving better performance than traditional machine learning algorithms (Zhang et al., 2018). We explore both monolingual and multilingual learning paradigms. Multilingual approaches enable us to use both English and Spanish datasets for training.

The most popular input features in deep learning are word embeddings. Embeddings are fixed length vectors with real numbers as components, used to represent words in a numeric way. The input layers to our models consist of MUSE (Conneau et al., 2017) or ELMo (Peters et al., 2018) word embeddings.

MUSE embeddings are multilingual embeddings based on fastText. They are available in different languages, where the words are mapped into the same vector space across languages, i.e. words with similar meanings across languages have a similar vector representation.

ELMo provide a deep representation of words based on output of a three layer pre-trained neural network. The representation for a word is based on the context in which the word is used. However, they are not multilingual representations.

To work around the monolinguality of ELMo, we use a technique called *adversarial learning* (Ganin and Lempitsky, 2014). Adversarial networks consist of three parts:

- *Feature extractor* responsible for creating representations belonging to the same distribution regardless of input data distribution i.e. of the language the messages are in. This transformation is learned during training.
- *Classifier* responsible for the classification i.e. labeling hateful utterances.
- *Discriminator* responsible for predicting the language of a given message.

During backpropagation, the loss from classifier ($\mathrm{L_{cls}}$) is computed the standard way. The loss from discriminator ($\mathrm{L_{dis}}$) has its sign flipped and is multiplied by *adversarial lambda* (λ). The discriminator works *adversarialy* to the classificator.

$$Loss = L_{cls} - \lambda L_{dis} \tag{1}$$

The loss from the discriminator encourages the feature extractor to create indistinguishable representations for messages across languages. This is most often implemented by a gradient reversal layer.

2 Implementation details

2.1 Preprocessing

Traditionally, neural network models have a very simple preprocessing pipeline. However, internet communication is very bloated (URLs, mentions, emoji etc.). As such we have decided to remove all the noise from the messages.

At first, we remove URLs and name mentions from messages. These contain no useful information for our prediction. Afterwards, we transform malformed markup characters such as *&gt* into their one character representations (>). We also remove the hash symbol from hashtags as it can be problematic for tokenizers to work with. Next we employ demojization. We use a Python library called *Emoji*[1]. For example, this let us change the unicode representation of a thumbs up emoji into :thumbs_up:, which is then parsed into usable text 'thumbs up'. The next step is tokenization and stop words removal. For this step, we use a library called *spaCy*[2]. We chose this library as it has support for both English and Spanish and we aim to have the same preprocessing pipeline for different languages. We also remove lone standing non-alphanumeric characters, which are often found after tokenization. As the last few steps, we change all characters into lowercase, change numbers into a number token. Sentence size is limited to 64. This was enough for nearly all of the tweets after preprocessing.

2.2 Tested architectures

For MUSE, we use pretrained embeddings made available by Facebook research. We also use pretrained ELMo representations (Che et al., 2018; Fares et al., 2017), which support English as well as Spanish. Both can be found on GitHub [3] [4]. The embeddings were not modified during training.

We examine two different model architectures: LSTM based one and a CNN+LSTM hybrid. The combination of two learning paradigms, two model architectures and two different input representations sum up to 8 different models. All of the models use cross-entropy as the loss function.

2.2.1 Monolingual approaches

Monolingual models were used and trained independently on English and Spanish parts of the dataset.

LSTM-based approach

We use both word-level and char-level representations with ELMo. The representations are then independently fed into a bidirectional LSTM layer of size 64. The output of each of these layers is then fed into an attention layer.

Next, the outputs are concatenated into a single vector and used as an input of a fully connected layer with 20 cells with ReLU activation function. The last layer is a softmax layer with L1 and L2 regularization used for final predictions. The output is then the probability of classes for predicted

[1]https://pypi.org/project/emoji/

[2]https://spacy.io/

[3]https://github.com/facebookresearch/MUSE

[4]https://github.com/HIT-SCIR/ELMoForManyLangs

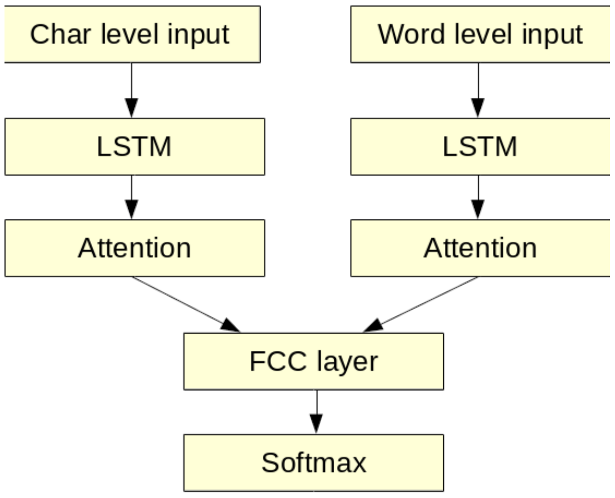


Figure 1: LSTM-based ELMo monolingual model

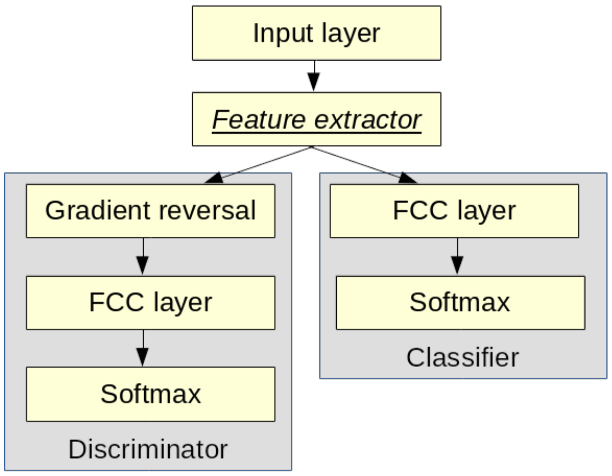


Figure 2: Base adversarial model

variable (non hate speech or hate speech). The model can be seen on Figure 1.

For MUSE, we have only word-level information available. As we have only one input, we only need one LSTM and attention layer. Otherwise, the models are the same.

CNN-based approach

The input layer is fed into a convolutional layer. This layer performs a 1d convolution with 100 filters and a kernel size of 4 with a relu activation function. This is then max pooled with a pool size of 4 and stride of 4. These layers can be understood as a feature extrator part of the model. These extracted features are then fed into a monodirectional LSTM layer with size of 64. The output is global max pooled and fed into the last softmax layer. For ELMo we have used the average representation of all its layers.

2.2.2 Multilingual approaches

Multilingual models were trained on concatenated English and Spanish data.

Multilingual MUSE models

With MUSE embeddings a multilingual approach is straightforward. We use both the approaches previously mentioned (LSTM and CNN+LSTM) without any further changes, as they are implicitly multilingual.

Multilingual ELMo models

The base architecture of our model can be seen on Figure 2. After the input layer is a feature extractor. We have used either an LSTM with attention or a 1d convolutional layer with max-pooling as described in previous sections. The discriminator and classifier include a single FCC layer with a final softmax layer in both cases. The FCC layers have 32 cells each.

The difference between them is the presence of a gradient reversal layer in the discriminator. The gradient is multiplied by -0.25 during backpropagation. This value for adversarial lambda was found empirically. Both the classifier and discriminator were trained simultaneously.

Architecture	Acc	Rec	Prec	F1
$\text{LSTM}_{\text{mono-elmo}}$	**0.733**	0.676	0.697	**0.683**
$\text{CNN}_{\text{mono-elmo}}$	0.69	0.698	0.640	0.655
$\text{LSTM}_{\text{mono-muse}}$	0.675	0.694	0.606	0.645
$\text{CNN}_{\text{mono-muse}}$	0.658	**0.761**	0.577	0.655
$\text{LSTM}_{\text{multi-elmo}}$	0.695	0.386	**0.799**	0.517
$\text{CNN}_{\text{multi-elmo}}$	0.673	0.448	0.693	0.52
$\text{LSTM}_{\text{multi-muse}}$	0.664	0.677	0.594	0.632
$\text{CNN}_{\text{multi-muse}}$	0.661	0.677	0.59	0.632

Table 1: Results on English dataset (Task A)

3 Results evaluation

We show detailed results in both English (Table 1) and Spanish (Table 2). We use a subscript of *mono* or *multi* to differentiate between learning methods and *muse* or *elmo* to differentiate between architectures in the table. The table was completed by computing the mean of 5 runs of each model on the validation part of the datasets. The validation set consisted of 10% available data. Multilingual models were trained with concatenated English and Spanish datasets.

None of the multilingual models were able to

Architecture	Acc	Rec	Prec	F1
$\text{LSTM}_{\text{mono-elmo}}$	**0.768**	**0.742**	**0.748**	**0.738**
$\text{CNN}_{\text{mono-elmo}}$	0.726	0.65	0.726	0.657
$\text{LSTM}_{\text{mono-muse}}$	0.711	0.712	0.662	0.689
$\text{CNN}_{\text{mono-muse}}$	0.72	0.731	0.673	0.699
$\text{LSTM}_{\text{multi-elmo}}$	0.556	0.123	0.419	0.173
$\text{CNN}_{\text{multi-elmo}}$	0.588	0.332	0.712	0.345
$\text{LSTM}_{\text{multi-muse}}$	0.723	0.701	0.684	0.692
$\text{CNN}_{\text{multi-muse}}$	0.718	0.688	0.681	0.685

Table 2: Results on Spanish dataset (Task A)

outperform the baseline monolingual LSTM based model with ELMo. Not even in a multilingual setting of averaging results between languages. Multilingual MUSE has not shown any significant increase in performance compared to monolingually trained MUSE.

The results show how potent ELMo embeddings are. Online content can often be offensive and vulgar, while still being non-hateful. This is often enough for a model to classify an utterance as hate speech (Davidson et al., 2017; Hemker, 2018). In these situations, ELMo has an advantage, as the representations are built entirely in the context of a sentence as a whole.

The adversarial models achieved the worst performance. On first glance, judging by accuracy, the models seem to perform on a very average level. After further analysis, we can see that their performance was very poor and inconsistent, e.g.the LSTM based model achieved only 0.123 recall on spanish dataset. The model labeled only a few messages as hate speech and even those not very successfully. The relatively high accuracy was a result of data distribution, as 55.6% of the data was non-hate speech.

We can also see that only in this category the CNN based models outperformed LSTM based models. This implies that for adversarial learning to work, one has to use a very robust feature extractor. It is also the only time that the performance on English was higher than on Spanish. This is the result of data scarcity, as the extractor had a hard time creating truly multilingual representations. This could also be seen during training as the discriminator hovered around 90% accuracy.

For our task submission, we have used the monolingual LSTM model based on ELMo, which we considered as our baseline model. We have achieved results shown in Table 3.

Language	Acc	Pre	Rec	F1	Place
English	0.47	0.59	0.54	0.42	44
Spanish	0.71	0.7	0.7	0.7	18

Table 3: Results on test dataset

4 Conclusion and future work

In this paper we have evaluated a few simple neural network models in a monolingual and multilingual context. We have included our unsuccessful models to inspire further research in this direction.

We conclude that the quality of word representations used has a significant impact on the performance of a model. Changing between MUSE and ELMo resulted in a 3 - 4% increase in accuracy even when MUSE based models could benefit from multilingual training. The contextual nature of ELMo representations make them much more flexible and less domain constrained than traditional word embeddings. Simple models (as the one we proposed) are able to achieve decent results this way. We can also see that using adversarial learning needs a lot of available data to be at all viable.

We believe that more research should be put into multilingual solutions. The feature extractor needs more training data to create truly ambiguous representations of utterances between languages. We will look into testing our model with more training data to evaluate the value of adversarial learning for multilingual hate speech detection or pretraining the feature extractor on a different task with more data available.

References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics", location = "Minneapolis, Minnesota.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages

55–64, Brussels, Belgium. Association for Computational Linguistics.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language.

Murhaf Fares, Andrey Kutuzov, Stephan Oepen, and Erik Velldal. 2017. Word vectors, reuse, and replicability: Towards a community repository of large-text resources. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 271–276, Gothenburg, Sweden. Association for Computational Linguistics.

Yaroslav Ganin and Victor Lempitsky. 2014. Unsupervised Domain Adaptation by Backpropagation. (i).

Konstantin Hemker. 2018. Data Augmentation and Deep Learning for Hate Speech Detection.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive Language Detection in Online User Content. *Proceedings of the 25th International Conference on World Wide Web - WWW '16*, pages 145–153.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection using Natural Language Processing. *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, (2012):1–10.

Zeerak Waseem. 2016. Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 138–142. Association for Computational Linguistics.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech Using a Convolution-GRU Based Deep Neural Network. 7185(June).

Saagie at Semeval-2019 Task 5: From Universal Text Embeddings and Classical Features to Domain-specific Text Classification

Miriam Benballa
Saagie / LITIS
Rouen, France
miriam.benballa@saagie.com

Sebastien Collet
Saagie
Rouen, France
sebastien.collet@saagie.com

Romain Picot-Clemente
Saagie
Rouen, France
romain.picotclemente@saagie.com

Abstract

This paper describes our contribution to SemEval 2019 Task 5: Hateval. We propose to investigate how domain-specific text classification task can benefit from pretrained state of the art language models and how they can be combined with classical handcrafted features. For this purpose, we propose an approach based on a feature-level Meta-Embedding to let the model choose which features to keep and how to use them.

1 Introduction

In this paper, we describe our system for Task 5 of SemEval 2019 (Basile et al., 2019), namely Multilingual detection of hate speech against immigrants and women in Twitter (HatEval). In this task, participants were asked to automatically classify English and Spanish tweets as hateful or not for Subtask A, and to predict if these tweets are aggressive or not, then identify whether the target is generic or individual for Subtask B. We participated in all subtasks for both English and Spanish.

Our main interest in this competition is to evaluate how a domain-specific dataset can take advantage of unsupervised data and moreover, how very different features can be combined efficiently in a deep neural network to improve classification. For this purpose, we propose to exploit state of the art pretrained deep learning models in text classification and classical features into an architecture that allows combining them dynamically.

Our work consists of three steps: features creation, dynamic meta-embedding and finally combining this information to classify tweets. The next sections are organized as follows: in section 2, we will briefly cover the related work, in section 3 we will explain our model, then in section 4 we will expose our experiences, and finally we will introduce our results in section 5.

2 Related Work

A successful classical approach for tweets classification and sentiment analysis is to use neural networks on top of pre-trained word embeddings. Word embeddings are trained with unsupervised data with a method called distant supervision (Go et al., 2009). Deriu et al. (2016) use Convolutional neural networks on top of those word embeddings while Cliche (2017) is using an ensemble of CNNS and LSTMs. Both solutions won respectively SemEval task 4 in 2016 and 2017.

For tasks more closely related to SemEval Task 5, Sánchez Gómez (2018) won the IberEval 2018 Aggressiveness detection task with an Ensembling of several SVMs models. The Ensembling is done with a Genetic Algorithm. Cuza et al. (2018) propose a model with a Bi-LSTMs network with attention layers on top of pre-trained word embeddings. Their solution got the 2nd place.

On the Mysogyny detection task in IberEval 2018, Pamungkas et al. (2018) won with an SVM trained on a lot of handcrafted features. They used stylistic, structural and lexical features to represent information such as Hashtag Presence, Link Presence, Swear Word Count, Swear Word Presence, Sexist Slurs Presence and Woman-related Words Presence. SemEval 2019 Task 5 is a combination of those two IberEval 2018 tasks.

However, a recent trend in Natural Language Processing has been the use of Transfer Learning from universal sentence embedders to tackle text classification tasks such as Hate Speech detection. This approach is particularly useful when little supervised data is accessible.

The main goal of these universal sentence embedding methods is to embed a sentence in a fixed sized vector that encodes as best as possible the sentence semantic and syntactic information. There are various universal sentence embedding approaches such as the Skip-Thought Vectors (Kiros et al., 2015) that adapt the skip-gram Model of the original Word2Vec to the sentence level, or Infersent (Conneau et al., 2017) that uses a model trained in a supervised fashion on a Natural Language Inference Task.

However, the most promising approaches are probably those based on language models. OpenAI (Radford et al., 2018) propose such a solution called GPT based on the Transformer architecture (Vaswani et al., 2017). In their work, a Transformer is trained in a generative unsupervised manner on a Language Modeling task. The model tries to continuously predict the following word of a text given the rest of the text. Another approach, BERT (Devlin et al., 2018) is also based on the Transformer architecture, but the unsupervised learning scheme is a bit different. The idea is to counter the left-right bias that may arise with classical language modeling. During the training phase, the model tries to predict words hidden randomly in the text and it also tries to tell whether two sentences are following each other or not. These models are trained on datasets such as Wikipedia and BooksCorpus (Zhu et al., 2015).

Both approaches give good results on the GLUE benchmark (Wang et al., 2018), which is a language understanding benchmark based on a diverse range of NLU tasks. Models that present high scores on this benchmark should have a good Transfer Learning capability.

Since the emergence of Word Embeddings with the Word2Vec (Mikolov et al., 2013) in 2013, numerous Word Embeddings approaches were developed such as Glove (Pennington et al., 2014), FastText (Bojanowski et al., 2017) or more recently Elmo (Peters et al., 2018). Evaluating the quality of such Word Embedding in a fair manner is a difficult task and these embeddings approaches may perform best in various situations. Dynamic Meta-Embeddings (Kiela et al., 2018) is a sentence representation method that lets a neural network figure out which Word Embedding from an ensemble to use depending on the situation.

3 Model Description

Universal sentence embedding is a way to share knowledge across different tasks. It is particularly helpful in situations with very small training dataset such as SemEval2019 Task 5 (10000 tweets in the training and development set). A pretrained sentence embedding model aims at a general syntactic and semantic understanding of the tweets.

However, the vocabulary and expressions used in this task are really context-specific so it seems necessary to be able to bring some of this specific content into the universal model. Moreover, we argue each sentence representation can potentially bring additional information to the others. Hence, instead of selecting the best sentence representation for our task, we propose to let a model find the best combination of multiple sentence representations with a Dynamic Meta Embedding approach. This latter works as follows.

From a sentence s, we have n sentence embedding types with different length d_i, leading to a set $\{s_i\}_{i=1}^{n} \in \mathbb{R}^{d_i}$.

Similarly to (Kiela et al., 2018), each sentence embedding is projected to a same d'-dimensional space with a learned linear function $s'_i = P_i s_i + b_i$. where $P_i \in \mathbb{R}^{d' \times d_i}$. These projections are then combined with a weighted sum

$$s_i^{final} = \sum_{i=1}^{n} \alpha_i s'_i$$

where $\alpha_i = g(s'_i)$ are scalar weights which depend on projected sentence embeddings s'_i:

$$\alpha_i = g(s'_i) = \phi(a \cdot s'_i + b)$$

where $a \in \mathbb{R}^{d'}$ and $b \in \mathbb{R}$ are learned parameters and ϕ is a softmax function, so that $\sum_{i=1}^{n} \alpha_i = 1$.

All α_i can be seen as importance weights. When averaging them on all the train dataset, they can be exploited to select important features representations.

For embedding sentences, we propose to use state of the art pretrained models: Bert and GPT. Since they are general sentence embeddings, we finetuned them on our specific tasks to get more specific embeddings (we also tried without finetuning them but got very poor classification results).

Beside these sentence embeddings, we created several classical sentence representations. We constructed all the features suggested by Pamungkas et al. (2018) (see the paper for more details) and some extra features as follows:

- ***Language Model Perplexity*** Perplexity score of each tweet according to the language model kenlm[1] ;

[1] https://github.com/kpu/kenlm

- ***BayesianEncodingHashtag*** Probability of hashtag according to the target class ;
- ***hashtagUrlPresence*** One-Hot encoding on presence of urls and hashtags in tweets ;
- ***Abreviation*** Abreviation counting from a custom lexicon ;
- ***BagOfPOSTagging*** Counting the different POS tags in each tweet ;
- ***NMF*** Non-negative Matrix Factorization on the co-occurrence matrix of words ;
- ***LDA*** Latent Dirichlet Allocation on the tweets ;
- ***BagOfEmojiFeatures*** One-Hot encoding on presence of emojis in tweets ;
- ***nbWords*** Number of words in each tweet, normalized by mean ;
- ***Textstat*** Readability features according to the python package textstat[2] ;
- ***nbChar*** Number of characters in each tweet, normalized by mean.

However, the importance weights from the dynamic weighted sum of our model show that most of these representations were not of interest for the predictions, and were reducing the F1-score. Hence, we made a feature selection based on these weights for each subtask. In the next subsections, we detail the different sentence representations we used for each subtask.

3.1 Pre-processing

We didn't use a lot of pre-processing besides lowercasing, in order to benefit from the representations capabilities of BERT and GPT. These models are using BPE encoding (Sennrich et al., 2015), so the models are based on subword units and not on words. This way, out of vocabulary words such as those with spelling mistakes or very context-specific ones may still be processed in a useful way by the model. However, a kind of spelling mistake correction might have been useful. The main pre-processing scheme we used is the replacement of usernames and urls by a specific token.

We normalized the most frequent hashtags in order to keep only one spelling (for instance #buildthatwall and #buildthewall were processed to have the same spelling). We also processed the most frequent abbreviations by replacing them with their full form. Finally we tried a splitting words approach on the hashtags in order to help the BPE encoding to get sensible of subword units. This did not improve performance, so this preprocessing was not kept in our final submission.

3.2 Subtask A en: Hateful or not

This subtask consists in classifying each English tweet as hateful or non hateful. For each tweet, the following features have been selected and given as inputs to our model:

- Bert embeddings: 3 different finetuned pretrained Bert embeddings[3], one for each target class (HS, TR, AG). Leading to 3 sentence representations of 768 features.
- GPT embeddings: 3 different finetuned pretrained GPT embeddings[4], one for each target class (HS, TR, AG). Leading to 3 sentence representations of 768 features.
- Hate Word Count: Count the presence of words into a lexicon extracted from HateBase[5], leading to 1 feature
- Bag of Emojis: Count the presence of Emojis grouped by type, leading to 155 features (number of emojis)

3.3 Subtask B en: Target and Aggressivity

Subtask B consists in predicting in addition to the hate speech, the target of the hate speech (TR - a group or an individual) and the aggressiveness (AG). We used the same approach, architecture and features to predict the labels TR and AG. Each label is predicted independently. However, we added a simple post-processing correction based on the predictions we made for HS: if a tweet is classified as not hateful, we set the target to generic (TR prediction to 0) and labeled the tweet as not aggressive (AG prediction to 0). This rule has been deducted from the way tweets are labeled: non hateful tweets are always classified as generic and not aggressive.

3.4 Subtask A/B es

We used the same model for the Spanish dataset and translated Spanish tweets to English with machine translation. In doing this, we could employ the same type of features as we used for English.

[2] https://github.com/shivam5992/textstat

[3] https://github.com/huggingface/pytorch-pretrained-BERT

[4] https://github.com/huggingface/pytorch-openai-transformer-lm

[5] https://hatebase.org/

Dataset	Total	HS	TR	AG
Train EN	9000	3783	1341	1559
Dev EN	1000	427	219	204
Test EN	2970	1679	522	590
Train ES	4500	1857	108	1001
Dev ES	500	222	137	176
Test ES	1599	660	423	474

Table 1: Hate speech dataset.

For the subtask B, the same corrections were applied for TR and AG using HS predictions.

4 Experimental Setup

4.1 Data

For each language, a training, a development and a test set were provided. These datasets were manually annotated using Figur8[6] crowdsourcing platform. Statistics on label distribution can be found in Table 1.

4.2 Parameter settings

Our model is implemented in PyTorch[7] and trained on 2 GPU Tesla V100. For the finetuning of Bert and GPT, we used the default parameters of their respective repository but trained on 10 epochs.

For the learning of the Meta Embedding model, we used a batch size of 64 and Adam optimizer with a variable learning rate (the Noam decay introduced in Vaswani et al. (2017)). The dropout rate is set to 0.6. To avoid over-fitting issues and to be able to reproduce and compare our results, we used Scikit-learn[8] implementation of Stratified Shuffle Split, with 10 splits on the concatenated train and dev dataset. Our results metrics are the means of the values obtained on the 10 splits.

5 Results

This section presents the evaluation of the SemEval-2019 Task 5: HatEval. The official measure for this task was the macro F1-measure. Note that for the Subtask B, evaluation was based on two criteria (each dimension evaluated independently or jointly), however the final ranking was based solely on the second criteria (Exact Match Ratio on the three labels). More details about the evaluation system can be found in the task description paper (Basile et al., 2019).

We saved the best epoch model for each of the 10 splits and we used them to make our final prediction for the test dataset. Then we used our 10 models to classify each tweet: to predict a tweet as hateful, at least half (5) of the models have to agree with this class. The same goes for subtask B to predict TR and AG, with in addition the post-processing described in subsection 3.3. Macro F1-scores and EMR scores with this agreement rule on the English development splits and test datasets are respectively presented in Table 2 and Table 3. This latter is our final submission for the competition. We can see a surprising decreasing of macro F1-score for the predictions on the test dataset of about 35 points compared to the predictions on our experimentation splits. We discuss about this result in the next section.

Table 4 and Table 5 show the results on the Spanish datasets. We can see that the finetuned BERT model gives good results on the test dataset (3 points better in macro F1 than the leader on subtask A) whereas it was worse than the other models on the splits dataset on our experiments. Our hypothesis is that the other models may have overfitted on the train dataset (especially the GPT model). Our Meta-embedding model seems to have been penalized by the GPT overfitting.

6 Discussion

6.1 Unsuccessful approaches

During this competition, we experimented many additional methods that did not successfully improve the results:

- We created an important quantity of features manually as described in section 3. However, most of them were not useful for the prediction according to the weights extracted from our model. We suppose this is either because the features from finetuned BERT and GPT models are able to capture most of the information provided by the other features or because it is difficult to blend handcrafted features with the ones obtained from BERT and GPT.

- We tried others universal sentence embedding models besides GPT and BERT such as InferSent and ULMFiT (Howard and Ruder, 2018) but without very good results. As

[6]https://www.figure-eight.com/
[7]https://pytorch.org/
[8]https://scikit-learn.org/stable/

Model	HS (subtask A)	TR	AG	EMR (subtask B)
GPT	82.33%	82.27%	71.10%	66.74%
BERT	82.87%	82.88%	74.11%	68.61%
Meta-Embedding (our submission)	**84.16%**	**83.93%**	**75.01%**	**70.55%**

Table 2: Mean results on English development splits.

Model	HS (subtask A)	TR	AG	EMR (subtask B)
GPT	**51.50%**	**73.78%**	**60.42%**	36.20%
BERT	48.84%	73.39%	59.16%	36.03%
Meta-Embedding (our submission)	49.60%	72.40%	57.80%	**37.40%**
Baseline SVM	45.00%	69.70%	58.70%	30.80%

Table 3: Final results on English test dataset.

SemEval task 5 is very specific vocabulary-wise, it is possible that universal models such as InferSent and ULMFit were not trained on enough data to provide good features representations of the tweets.

- For the Spanish dataset, we also tried the BERT Multilingual model which was released during the competition, but we also had lower results than using translation.

- We tried augmenting the dataset with external resources, especially with a similar labeled dataset[9] of tweets with hate speech and offensive language. Nevertheless, this method was decreasing the results, probably due to different labeling rules.

- Inspired from the back-translation proposed in (Edunov et al., 2018), we augmented the dataset by automatically translating each tweet to an other language (French, Spanish, Chinese) and back translated it to the initial language (English). This method can be assimilated to a transfer learning approach that should bring more variability in the dataset, and should improve the generalization ability of the model. Our tests did not show relevant improvement in F1-score but were decreasing the variance. Nevertheless, we did not develop enough this approach to conclude on its potential benefits.

- About our post-processing choice for Task B, we based it on the fact that our model for HS prediction was better than the models for TR and AG. Considering how much the performance on the HS task decreased on the test set compared with the decreasing on the TR and AG tasks it was probably not the best choice. A multi-label model might have been useful for this task considering the evaluation metric (each label prediction should not be independent).

- Finally, we tried to train our model on both English and translated Spanish datasets, but that did not improve our results.

[9]https://github.com/t-davidson/hate-speech-and-offensive-language/tree/master/data

6.2 About the testing set

The previous section shows an important difference on HS in terms of prediction quality (F1-score) between the development and the test datasets. This score difference seems to be experienced by every participant according to the development and test leaderboards. It seems that the test dataset contains a lot more of difficult tweets to classify in comparison with the train and development datasets. Our hypothesis is that the test dataset has not been collected like the other datasets (train and development) or that data were sorted in a particular way after the collection, which could explain such results.

In this setup it is interesting to see that the features extracted from the finetuned GPT generalize a little better (with a HS F1-score of 51.50) than our submited model (49.6 HS F1-score). Adding more features might have induced more overfitting on the training set.

Since the end of the competition, the state of the art on Natural Language Understanding on the GLUE Benchmark is a new model. It is a Multi-

Model	HS (subtask A)	TR	AG	EMR (subtask B)
GPT	81.67%	86.17%	79.77%	70.10%
BERT	78.21%	81.63%	75.48%	65.22%
Meta-Embedding (our submission)	**83.19%**	**86.79%**	**81.01%**	**74.98%**

Table 4: Mean results on Spanish development splits.

Model	HS (subtask A)	TR	AG	EMR (subtask B)
GPT	66.42%	74.17%	67.66%	51.53%
BERT	**76.88%**	**81.08%**	**76.55%**	**65.85%**
Meta-Embedding (our submission)	71.70%	80.90%	76.00%	63.50%
Baseline SVM	70.10%	78.10%	72.60%	60.50%

Table 5: Final results on Spanish test dataset.

Task Model based on BERT (Liu et al., 2019). It seems that the Multi-Task Learning approach improves the universality of BERT. We think that such a model could also improve our architecture on this task because a model trained in a Multi-Task manner should in theory be more robust to overfitting.

7 Conclusion

In this work, we investigated how a model could merge features obtained from unsupervised language models such as GPT and BERT with domain specific hand-crafted features. We presented an approach based on a feature-level Meta-Embedding to let the model choose which features to keep and how to use them. Our method systematically outperforms models based only on BERT or GPT features on our evaluation datasets, however it is not always the case on the test datasets. For instance, on the Spanish test dataset, BERT alone gives better results and on the English test dataset subtask A, GPT slightly outperforms our submission.

Our idea was that the data used for SemEval 2019 Task 5 is very domain-specific and present a peculiar vocabulary. We thought that universal sentence embeddings methods would not work very well since such vocabulary was probably not present during their unsupervised training and the sentence quality is also probably different. However, our results tend to show that it is not the case. For instance, a model using only BERT features would have been 1st on the Spanish task A. The BPE used as a pre-processing for these models is probably helping to deal with out-of-vocabulary words. On top of that, it seems that big unsupervised language models are able to learn data representation that generalize really well to unseen domains.

Acknowledgments

This work is part of a French RAPID project called SAPhIRS, in collaboration with LITIS Laboratory and Airbus Defence Space. The authors would like to thank Alexandre Pauchet and Simon Bernard for their help.

References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Mathieu Cliche. 2017. Bb_twtr at semeval-2017 task 4: twitter sentiment analysis with cnns and lstms. *arXiv preprint arXiv:1704.06125*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Carlos Enrique Muniz Cuza, Gretel Liz De la Pena Sarracén, and Paolo Rosso. 2018. Attention mechanism for aggressive detection. In *CEUR Workshop Proceedings*, volume 2150, pages 114–118.

Jan Deriu, Maurice Gonzenbach, Fatih Uzdilli, Aurelien Lucchi, Valeria De Luca, and Martin Jaggi. 2016. Swisscheese at semeval-2016 task 4: Sentiment classification using an ensemble of convolutional neural networks with distant supervision. In *Proceedings of the 10th international workshop on semantic evaluation*, CONF.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12).

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146.

Douwe Kiela, Changhan Wang, and Kyunghyun Cho. 2018. Dynamic meta-embeddings for improved sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1466–1477.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Endang Wahyu Pamungkas, Alessandra Teresa Cignarella, Valerio Basile, Viviana Patti, et al. 2018. 14-exlab@ unito for ami at ibereval2018: Exploiting lexical knowledge for detecting misogyny in english and spanish tweets. In *3rd Workshop on Evaluation of Human Language Technologies for Iberian Languages, IberEval 2018*, volume 2150, pages 234–241. CEUR-WS.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Claudia Nallely Sánchez Gómez. 2018. Ingeotec at mex-a3t: Author profiling and aggressiveness analysis in twitter using μtc and evomsa. *OPENAIRE*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

SINAI at SemEval-2019 Task 5: Ensemble learning to detect hate speech against inmigrants and women in English and Spanish tweets

Flor Miriam Plaza-del-Arco, M. Dolores Molina-González, M. Teresa Martín-Valdivia, L. Alfonso Ureña-López

Department of Computer Science, Advanced Studies Center in ICT (CEATIC)
Universidad de Jaén, Campus Las Lagunillas, 23071, Jaén, Spain
{fmplaza, mdmolina, maite, laurena}@ujaen.es

Abstract

Misogyny and xenophobia are some of the most important social problems. With the increase in the use of social media, this feeling of hatred towards women and immigrants can be more easily expressed, therefore it can cause harmful effects on social media users. For this reason, it is important to develop systems capable of detecting hateful comments automatically. In this paper, we describe our system to analyze the hate speech in English and Spanish tweets against Immigrants and Women as part of our participation in SemEval-2019 Task 5: hatEval. Our main contribution is the integration of three individual algorithms of prediction in a model based on *Vote* ensemble classifier.

1 Introduction

With the growing prominence of social media like Twitter or Facebook, more and more users are publishing content and sharing their opinions with others. Unfortunately, the content often contains hate speech language that can have damaging effects on social media users. This fact concerns to social media platforms like Facebook since according to an EU's report, it removes 82 percent of illegal hate speech on the platform, up from 28 percent in 2016[1].

Normally, hate speech can be aimed at a person or a group base on some characteristic such as race, sexuality, color, ethnicity, physical appearance, religion, among others (Erjavec and Kovačič, 2012). Currently, two of the targets most affected by these types of offensive comments are immigrants and women (Waseem and Hovy, 2016). In particular, when the hate speech is gender-oriented, and it specifically targets women, we refer to it as misogyny (Manne, 2017) and when the hate speech is against immigrants, we refer to it as xenophobia (Sanguinetti et al., 2018).

Recently, a growing number of researchers have started to focus on studying the task of automatic detection of hateful language online (Fortuna and Nunes, 2018; Fersini et al., 2018b), moreover, some academic events and shared tasks have taken place focusing on this issue (Fersini et al., 2018a). It is consider as a difficult task for social media platforms. For example, popular social media such as Twitter, Instagram or Facebook are not able to automatically solve this problem and depend on their community to report hateful speech content.

The severe consequences of this problem, combined with the large amount of data that users publish daily on the Web, requires the development of algorithms capable of automatically detecting inappropriate online remarks.

In this paper, we describe our participation in SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter (hatEval) (Basile et al., 2019). In particular, we participate in task A in English and Spanish. It is a binary classification task and the objective is predict whether a tweet with a given target (women or immigrants) is hateful or not hateful.

The rest of the paper is structured as follows. In Section 2 we explain the data used in our methods. Section 3 presents the details of the proposed systems. In Section 4, we discuss the analysis and evaluation results for our system. We conclude in Section 6 with remarks and future work.

2 Data

To run our experiments, we used the Spanish and English datasets provided by the organizers in SemEval19 Task 5 : HatEval (Basile et al., 2019). The datasets contain tweets with several fields. Each tweet is composed for an identifier (id), the

[1]https://cnb.cx/2RGmEwel

text of the tweet (text), the mark of hate speech (HS), being 0 if the text is not hateful and 1 if the text is hateful, the mark of recipient of text (TR), being 1 if the target is a single human and 0 if the target is a group of persons and the last field (AG) is the mark that identifies if the text is aggressive whose value is 1, else 0 in the case opposite. During pre-evaluation period, we trained our models on the train set, and evaluated our different approaches on the dev set. During evaluation period, we trained our models on the train and dev sets, and tested the model on the test set. Table 1 shows the number of tweets used in our experiments for Spanish and English.

Dataset	train	dev	test
Spanish	4,500	500	1,600
English	9,000	1,000	3,000

Table 1: Number of tweets per HatEval dataset

We only take into account the fields text and HS for our experiments because we participate in task A in English and Spanish.

3 System Description

In this section, we describe the systems developed for the Hateval task 5, subtask A in English and Spanish.

3.1 Our classification model

In first place, we preprocessed the corpus of tweets provided by the organizers. We applied the following preprocessing steps: the documents were tokenized using NLTK library [2] and all letters were converted to lower-case. In second place, an important step is converting sentences into feature vectors since it is a focal task of supervised learning based sentiment analysis method. Therefore, our chosen statistic feature for the text classification was the term frequency (TF) taking into account unigrams and bigrams because it provided the best perfomance.

During our experiments, the scikit-learn machine learning in Python library (Pedregosa et al., 2011) was used for benchmarking. Our classification model based on *Vote* ensemble classifier combined three individual algorithms: *Logistic Regression (LR)*, *Decision Tree (DT)* and *Support Vector Machines (SVMs)*. We have tested with other models such as *naive bayes* and *multilayer perceptron* but we have obtained better results with the combination of the three algorithms mentioned above. In Figure 1, it can be seen our model. We train our model with the train and dev set and we evaluated it with the test set. There are many combinations to implement a model when we apply different classifiers with several parameters. Therefore, one of the most important step was to find the best individual classifiers for the problem. After doing several experiments with each classifier independently, we came up with SVMs, LR and DT classifiers. In order to improve the performance of each classifier, we choose the best optimization of the parameters in each of them.

[2] https://www.nltk.org/

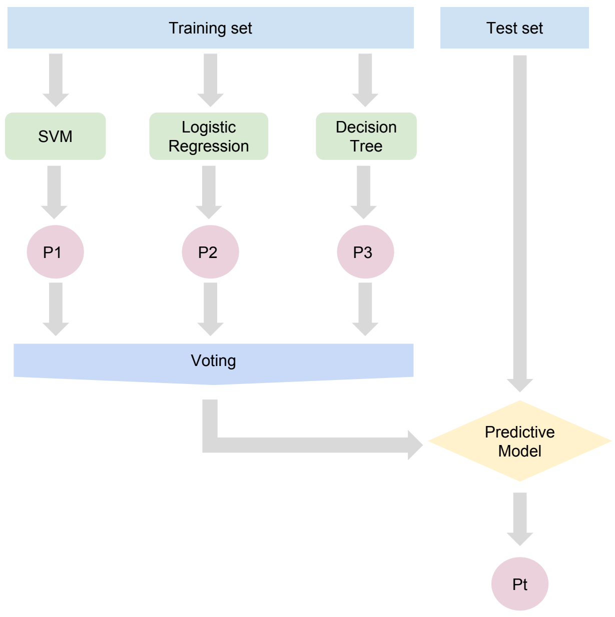


Figure 1: Systems architecture.

3.2 Classifiers

1. *Logistic Regression* is an statistical method for prediction binary classes. It computes the probability of an event occurrence utilizing a logit function. In order to optimize the parameters of LR in our English and Spanish experiments, we used the penalty parameter equal to l1 regularization.

2. *Decision Tree* is a flowchart-like tree structure where an internal node represents features, the branch represents a decision rule, and each leaf node represents the outcome. In order to optimize the parameters of DT in our English and Spanish experiments, we leave

User name (r)	Test			
	P	R	F1	Acc
francolq2 (1)	0.734	0.741	0.73	0.731
luiso.vega (2)	0.729	0.736	0.73	0.734
fmplaza (14)	**0.707**	**0.713**	**0.707**	**0.711**
SVC baseline (21)	0.701	0.707	0.701	0.705
DA-LD-Hildesheim (40)	0.493	0.494	0.493	0.511

Table 2: System Results per team in subtask A of hatEval task in Spanish.

User name (ranking)	Test			
	P	R	F1	Acc
saradhix (1)	0.69	0.679	0.651	0.653
amontejo (5)	0.601	0.577	0.519	0.535
SVC baseline (35)	0.595	0.549	0.451	0.492
fmplaza (40)	**0.627**	**0.555**	**0.443**	**0.493**
sabino (71)	0.652	0.521	0.35	0.447

Table 3: System Results per team in subtask A of hatEval task in English.

the default parameters.

3. *Support Vector Machines* is a linear learning technique that finds an optimal hyperplane to separate our two classes (hateful and not hateful speech). Many researchers have reported that this classifier is perhaps the most accurate method for text classification (Moraes et al., 2013) and also is widely used in sentiment analysis (Tsytsarau and Palpanas, 2012). In order to optimize the parameters of SVMs in our English and Spanish experiments, we used the parameter C equal to 0.6 and the kernel used was linear.

4. *Vote* is one of the most straightforward ensemble learning techniques in which performs the decision process by applying several classifiers. Voting classifier combines machine learners by using a majority vote or predicted probabilities for the classification of samples. The predictions made by the submodels can be assigned weights. In our case, the weights are distributed as follows: 2 for LR and SVM and 1 for DT.

4 Experiments and analysis of results

During the pre-evaluation phase we carried out several experiments and the best experiments were taken into account for the evaluation phase. The system has been evaluated using the official competition metrics, including Accuracy (Acc), Precision (P), Recall (R) and F1-score (F1). The metrics have been computed as follows:

$$P = \frac{\text{number of correctly predicted instances}}{\text{number of predicted labels}} \tag{1}$$

$$R = \frac{\text{number of correctly predicted labels}}{\text{number of labels in the gold standard}} \tag{2}$$

$$F1 = \frac{2 * P * R}{P + R} \tag{3}$$

$$Acc = \frac{\text{number of correctly predicted instances}}{\text{total number of instances}} \tag{4}$$

The results of our participation in the subtask A of hatEval task during the evaluation phase can be seen in Table 2 for Spanish and in Table 3 for English.

In relation to Spanish results, it should be noted that we achieve a high position in the ranking outperforming the baseline result. Our position in the ranking is 14th of 41 participating teams. Therefore, we consider that the chosen individual classifiers in the voting system are appropriate to build the metaclassifier.

Therefore, our chosen statistic feature for the text classification was the term frequency (TF) taking into account unigrams and bigrams because it provided the best perfomance. One important feature to consider is the use of bigrams in TF, because during the pre-evaluation phase we noted that our results outperformed when we took into account the bigrams comparing it only to the unigrams.

In relation to English results, using the same system as for Spanish we achieved worse results and we did not outperform the baseline. However, we are ranked 40th out of 71 participating teams.

5 Conclusions

In this paper, we present the system we developed for our participation in SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter (hatEval). Specially, we have participated in subtask A in Spanish and English.

Our system was developed focus on Spanish. Therefore, we achieve better results in this language. On the one hand, one of the reasons could be the different employment of misogynistic or xenophobic words in one language with respect to the other (Canós, 2018). For example, the word "puta" in Spanish, can be consider a misogynistic word or in a bigram like "puta madre" can be similar to the word "fantastic". On the other hand, the way to insult women is not the same as the way to insult immigrants. For these reasons, systems make mistakes and should be considered different systems for these targets (immigrants and women).

Another important issue is that the participation in Spanish subtask is lower than the participation in English subtask. For this reason, we will continue developing systems in Spanish since it is one of the most spoken languages in the world and we consider a very challenging task.

Acknowledgments

This work has been partially supported by Fondo Europeo de Desarrollo Regional (FEDER) and REDES project (TIN2015-65136-C2-1-R) from the Spanish Government.

References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Jose Sebastián Canós. 2018. Misogyny identification through svm at ibereval 2018.

Karmen Erjavec and Melita Poler Kovačič. 2012. "you don't understand, this is a new war!" analysis of hate speech in news web sites' comments. *Mass Communication and Society*, 15(6):899–920.

Elisabetta Fersini, Debora Nozza, and Paolo Rosso. 2018a. Overview of the evalita 2018 task on automatic misogyny identification (ami). *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA18), Turin, Italy. CEUR. org*.

Elisabetta Fersini, Paolo Rosso, and Maria Anzovino. 2018b. Overview of the task on automatic misogyny identification at ibereval 2018.

Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Kate Manne. 2017. *Down girl: The logic of misogyny*. Oxford University Press.

Rodrigo Moraes, JoãO Francisco Valiati, and Wilson P GaviãO Neto. 2013. Document-level sentiment classification: An empirical comparison between svm and ann. *Expert Systems with Applications*, 40(2):621–633.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Manuela Sanguinetti, Fabio Poletto, Cristina Bosco, Viviana Patti, and Stranisci Marco. 2018. An italian twitter corpus of hate speech against immigrants. In *Language Resources and Evaluation Conference-LREC 2018*, pages 1–8. ELRA.

Mikalai Tsytsarau and Themis Palpanas. 2012. Survey on mining subjective data on the web. *Data Mining and Knowledge Discovery*, 24(3):478–514.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

SINAI-DL at SemEval-2019 Task 5: Recurrent networks and data augmentation by paraphrasing

Arturo Montejo-Ráez, Salud María Jiménez-Zafra,
Miguel Ángel García-Cumbreras, Manuel Carlos Díaz-Galiano
CEATIC / Universidad de Jaén
Las Lagunillas s/n
23071 Jaén (Spain)
`{amontejo, sjzafra, magc, mcdiaz}@ujaen.es`

Abstract

This paper describes the participation of the SINAI-DL team at Task 5 in SemEval 2019, called HatEval. We have applied some classic neural network layers, like word embeddings and LSTM, to build a neural classifier for both proposed tasks. Due to the small amount of training data provided compared to what is expected for an adequate learning stage in deep architectures, we explore the use of paraphrasing tools as source for data augmentation. Our results show that this method is promising, as some improvement has been found over non-augmented training sets.

1 Introduction

We have participated in SemEval 2019 Task 5, named *HatEval* (Basile et al., 2019), which encourage participants to identify hate speech in tweets. The small amount of training data provided makes difficult to train a deep architecture, so strategies like transfer learning and data augmentation are explored in our work. A trained model for word embeddings in the two languages targeted by the tasks have been considered as transfer learning approach. Paraphrasing the tweets has also been tested for data augmentation, doubling the number of tweets available for training the network.

Our results are promising for English, but no improvements have been found for Spanish. Further analysis on the results and the quality of the paraphrasing tools used is needed, but the scores obtained in English encourage us to consider paraphrasing as a promising help in deep learning for natural language processing.

The paper is organized as follows: Section 2 introduces the two main strategies used to train the neural network: data augmentation and transfer learning. In Section 3, task data is analyzed in order to define hyperparameters values. Section 4 describes the neural network architecture applied. Section 6 gives more details on the paraphrasing approach used to generate more training data. Experiments and results are given in Section 7. Finally, Section 8 closes the contribution with some conclusions and proposals for future work.

2 Data augmentation and transfer learning

Nowadays, deep neural architectures are populating the scientific playground in many scenarios: image recognition speech recognition (Graves et al., 2013) and synthesis (Ze et al., 2013), and, of course, text classification (Zhang et al., 2015). But these supervised learning algorithms demands for valid use different requirements that sometimes are difficult to meet. One of the most difficult to overcame in some cases is the need for a large and varied learning dataset. When there is lack of data, two main strategies can be followed: *transfer learning* and *data augmentation*.

- Transfer learning. This approach proposes to train the network on different task that is learned over a large set of available data of similar nature. For example, train a language model over a Wikipedia dump. Then, last layers can be replaced by those that fit the target task, for instance, sentiment analysis, and then trained on the limited dataset for that task.

- Data augmentation. This approach is commonly used in image recognition, by applying different transformations over training images (rotation, shearing, blurring, mirroring...). In this way, we can augment the size of the training data in several orders of magnitude, making the learning process more feasible and robust.

We will use both of them in our system for hate speech detection.

3 Data analysis

Organizers provided data consisting of a set of tweets annotated as hateful (1) or not hateful (0) -*HS* label-, with a person (1) or group (0) as target -*TR* label- and as aggresive (1) or not aggresive (0) -*AG* label-. The distribution of tweets per language and dataset is shown in Table 1.

Language	Train	Dev	Test	Total
EN	9,000	1,000	3,000	13,000
ES	4,500	500	1,600	6,600

Table 1: Distribution of tweets per dataset.

We analyzed training and development data to see whether the distribution of labels were similar, obtaining a positive result, which is desirable to guarantee the validity of the model fit. Figure 1 presents ring charts corresponding to the relative values of labels distribution across samples and datasets. Labels are represented using a binary codification of *HS*, *TR* and *AG* annotations. For example, 111 codification corresponds to hateful tweets towards a person and with aggressive content.

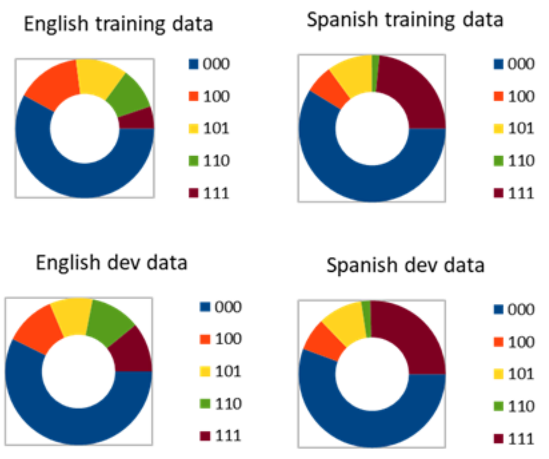


Figure 1: Datasets distribution per label.

In addition, we also analyzed the length of the tweets to decide which window size to use in our experiments. For this, a cumulative histogram for each dataset was generated according to different tweet lengths in order to select a size that would cover a high rate of tweets.

The sizes that cover 80 % and 90 % of tweets are summarize in Table 2 as quantiles 0.8 and 0.9 respectively. A value of 44 for quantile 0.8 means that 80 % of the tweets have a length of 44 or less. Taking into consideration the results we decided to select a window size of 40 words.

Data	Quantile 0.8	Quantile 0.9
train_EN	35	45
dev_EN	44	51
train_ES	37	46
dev_ES	38	49

Table 2: Length of tweets covering 80 % and 90 % of cases.

4 System description

We have implemented the proposed neural network using the Keras[1] library for Python, running on TensorFlow over a NVIDIA Titan X card. Each model took approximately 25 minutes to get trained and few seconds to classify development or test sets. The architecture of our neural network follows a sequence of layers as follows:

1. First layer: An embedding layer that is loaded with pre-trained weights, and converts each word into a 200-dimensional vector for English or a 300-dimensional one for Spanish.

2. Second layer: A bi-directional LSTM recurrent network with 512 activations and a dropout value of 0.5.

3. Third layer: A dense network with 128 activations and the *ReLU* function as activation function. A dropout of 0.5 is also applied after this network.

4. Fourth layer: last classification layer, with 3 activations on the sigmoid function, as we are in a multi-label classification task.

The model has been trained with the hyperparameters values specified in Table 3.

The text have been preprocessed as follows:

1. Lower case is applied.

2. Hashtags are splitted into several tokens according to a camel case approach. For example, "*#MeToo*" is converted into the terms "*<BOH>me too <EOH>*".

[1]`http://keras.io`

Parameter	value
Batch size	512
Loss function	binary cross-entropy
Optimization algorithm	Adam
Sequence length	40 terms
No. Epochs	100

Table 3: Hyperparameters.

3. Mentions are replaced by the token <*MENTION*>.

4. Unknown terms (those not found in the embedding dictionary) are replaced by the token <*UNK*>.

5. A final token <*EOT*> is added at the end of the tweet.

5 Transfer learning

We have taken already trained word embeddings for the first layer, allowing the weights of the these foreign models to get retrained during the learning process. We have used pre-trained GloVe (Global Vectors for Word Representation) models (Pennington et al., 2014) for the to targeted languages, English[2] and Spanish[3]. These are the models of word embeddings that we have transferred to the first layer in our architecture:

- For English we have used the weights from the GloVe Twitter model provided by the Stanford NLP Group, which is built over 2 billion tweets (27B tokens, 1.2M vocab, uncased, 200-dimensional vectors, 1.42 GB download).

- For For Spanish we have downloaded the GloVe model of the Spanish Billion Word Corpus(Cardellino, 2016), which generated 855,380 vectors of 300 dimensions.

Although FastText (Bojanowski et al., 2017) is considered the state-of-the-art for word embeddings representation, as it considers character n-grams instead of whole word forms, we have opted for GloVe due to the amount of available pre-trained models.

6 Data augmentation

The main problem with the dataset mentioned in the previous section is that there is a strong class imbalance between the samples with labels *"000"* and the samples with a different labeling. As Figure 1 illustrates, most of the samples were labeled as not hateful tweets towards a group and with not aggressive content, meaning that this class is highly dominated. Class imbalance introduces two key limitations: firstly, significant differences between accuracy and recall for some classes; and secondly, many machine learning models are prone to overfit on the majority class.

There are a number of ways to counter class imbalance, such as down-sampling the majority class, up-sampling the minority, and other hybrid solutions.

For each tweet, our system expand the information using paraphrasing. To express the same message with different words, we applied a online tool like RewriterTools[4]. For instance, the paraphrase of the tweet *"EU's hailed migrant plan 'a road to Hell' Czech Republic refuses involvement"* was *"EU's hailed migrant layout 'a avenue to Hell' Czech Republic refuses involvement"*.

Different configurations were created with the test data and the system, with the aim of obtaining results and analyzing the behavior of the different modules.

7 Experiments and results

We have performed several experiments to find good hyperparameters, but also evaluated the two main strategies proposed in our approach: transfer learning and data augmentation. In order to verify how transfer learning is good enough, i.e. how the predefined weights for GloVe could be further adjusted or not, we have checked the performance of the model trained on the official training set and evaluated on the development dataset on. Results in Table 4 shows that a small but consistent improvement is obtained if weights can be readjusted.

Next, we have empirically evaluated how paraphrasing helps to produce a better model or not. On these experiments the embeddings are always trainable. The paraphrasing tools allowed us to double the number of tweets in the training dataset. Thus, for English we have 18,000 tweets and

[2]https://nlp.stanford.edu/projects/glove/

[3]https://github.com/uchile-nlp/spanish-word-embeddings

[4]https://www.rewritertools.com/paraphrasing-tool

train	eval	embeddings	F-Score
train_EN	dev_EN	fixed	0.697875
train_EN	dev_EN	trainable	0.707153
train_ES	dev_ES	fixed	0.770951
train_ES	dev_ES	trainable	0.781350

Table 4: Experiments on fixed/trainable embeddings weights.

train	eval	F-Score
train_EN	dev_EN	0.707153
train_aug_EN	dev_EN	0.715593
train_ES	dev_ES	0.781350
train_aug_ES	dev_ES	0.767844

Table 5: Experiments on data augmentation

9,000 for Spanish. Table 5 shows the results obtained. Here different effects are noticiable. For English a slight improvement on macro F-Score metric is reported, and for Spanish the effect is very negative.

We have submitted predictions on the test set on models trained only on task B, so for task A we have submitted only predicted labels for HS column. For English, the training data has been the augmented official training set with paraphrased tweets. For Spanish, only the training tweets provided by the organizers have been used to produce the model. The official results obtained in this task are shown in Table 6

8 Conclusions and future work

Our proposal explores how transferred embeddings and data augmentation may help in a text classification task like HatEval. Paraphrashing does not report clear benefits. This can be due to the quality of the paraphrasing and the fact that new generated tweets are not very realistic. Other augmentation strategies could be explored, like forward-backward translation. We have found also the models trained exhibits high variance. That means that we are overfitting the model on training data, so despite the use of the dropout technique, early stopping, fewer parameters or more training data could help to produce a more robuts model. Another possible improvement is on how final labels are decided. Our system takes the final outputs of the last sigmoid layer as probabilities, so when the value is higher than 0.5 for a class, then the label is 1, 0 otherwise. We could try to set the thresholds using a SVM classifier on this sigmoid vector.

Subtask	F1 (avg)	EMR	rank/total
EN_A	0.519	-	5/70
EN_B	-	0.384	7/41
ES_A	0.686	-	26/39
ES_B	-	0.583	17/23

Table 6: Official HatEval results for our submissions.

Acknowledgements

This research work is partially supported by a grant from the Ministerio de Educación Cultura y Deporte (MECD - scholarship FPU014/00983), the project REDES (TIN2015-65136-C2-1-R) and a grant from the Fondo Europeo de Desarrollo Regional (FEDER).

References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Cristian Cardellino. 2016. Spanish Billion Words Corpus and Embeddings.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Heiga Ze, Andrew Senior, and Mike Schuster. 2013. Statistical parametric speech synthesis using deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7962–7966. IEEE.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

sthruggle at SemEval-2019 Task 5: An Ensemble Approach to Hate Speech Detection

Aria Nourbakhsh, Frida Vermeer, Gijs Wiltvank, Rob van der Goot
University of Groningen
Groningen, the Netherlands
{a.nourbakhsh, f.h.vermeer, g.g.wiltvank}@student.rug.nl
r.van.der.goot@rug.nl

Abstract

In this paper, we present our approach to detection of hate speech against women and immigrants in tweets for our participation in the SemEval-2019 Task 5. We trained an SVM and an RF classifier using character bi- and trigram features and a BiLSTM pre-initialized with external word embeddings. We combined the predictions of the SVM, RF and BiLSTM in two different ensemble models. The first was a majority vote of the binary values, and the second used the average of the confidence scores. For development, we got the highest accuracy (75%) by the final ensemble model with majority voting. For testing, all models scored substantially lower and the scores between the classifiers varied more. We believe that these large differences between the higher accuracies in the development phase and the lower accuracies we obtained in the testing phase have partly to do with differences between the training, development and testing data.

1 Introduction

An unwanted phenomenon that can be found across social media, is the publication of texts with hateful content. In SemEval-2019 Task 5 (Basile et al., 2019), it is defined as any communication that disparages a person or group. We focused on immigrants and women, which are two of the most targeted groups of people who are victims to this kind of discourse.

The micro-blogging service Twitter is a medium on which posts containing hateful content can be found in abundance. In order to filter out tweets with such content, machine learning and neural network techniques can be used to discriminate tweets which do contain hate speech from tweets which do not. Among the characteristics of Twitter data we can point out its noisiness and the use of emojis and hashtags, which can be taken into account for the classification task.

In this paper, we tried to solve this problem for English, by incorporating a variety of classification algorithms including Support Vector Machine (SVM), Random Forest (RF), and Bidirectional Long Short-Term Memory (BiLSTM). For the classical machine learning classification algorithms (SVM and RF) we used character n-grams and for the BiLSTM we used word embeddings trained on a huge amount of Twitter data. By combining these three models into an ensemble learning model, we achieved our best results on the development data, so we submitted this model for this shared task.

We start the paper by discussing some earlier work done in this field. Then, we describe the dataset we used for this task. In chapter 4, we present our approach. In chapter 5, we continue with the results of our methods and the discussion. Finally, we end with a conclusion and future work in chapter 7.

2 Related work

There has been done a lot of research regarding the automatic detection of hate speech on social media, in particular Twitter. A great deal of different approaches to solve this task had been implemented in different works. The majority of these studies was done on English texts. It is clear that there is quite some overlap between each of these approaches. However, direct comparison of previous approaches is not straightforward, as different datasets were used.

Most papers tried one or multiple different classifiers, albeit with different features, but in general SVM classifiers usually achieve the best performance (Saleem et al., 2016; Davidson et al., 2017).

Some papers divided the 'hate' class into two classes. For example, Watanabe et al. (2018) and Davidson et al. (2017) used the classes ‘offensive’ and ‘hate’, and Del Vigna et al. (2017) classified comments as ‘weak hate’ and ‘strong hate’.

Both the j48graft algorithm in Watanabe et al. (2018), and the SVM and LSTM in Del Vigna et al. (2017) performed better on a binary classification rather than a multiclass classification. Davidson et al. (2017) also tried different classifiers including Naive Bayes, decision trees, SVM and logistic regression. Their logistic regression and SVM classifiers achieved the best results. Waseem and Hovy (2016) tried different features for a logistic regression classifier, among which the character n-grams up to length of four in combination with the user’s gender information performed the best.

Other approaches are based on neural networks, like Zhang and Luo (2018). Their base convolutional neural network with a gap window (skipped CNN) had higher results than their SVM.

3 Data

The data provided by the organizers were collected from Twitter and manually annotated via the Figure Eight[1] crowdsourcing platform. All tweets contain a numeric ID, text of the tweet, and three labels with binary values (0 or 1). The first label indicates whether it is hate speech or not, the second if the target is a generic group of people or a specific individual and the third whether the tweeter is aggressive or not. The second and third labels can only be 1 if the first tag (hate speech or not) is 1 as well. However, we only used the first tag, since we only participated in task A, which is detecting hate speech.

The dataset for English contains 9,000 tweets for training, 1,000 for development and 3,000 for testing. Some tweets in the testing dataset were duplicates and removed from the dataset, resulting in 2,971 tweets. For each dataset, 57% of the tweets was labeled as non-hate speech and the rest as hate speech. For the testing phase, both training and development data were used for training the models.

[1]https://www.figure-eight.com/platform/

4 Method

4.1 Preprocessing

Before training the classifiers, we did some preprocessing steps over the data.

One of the reasons we did these preprocessing steps was that many words were not available in our word embeddings for the BiLSTM. Also, we could reduce the dimensionality of the character n-gram features for the RF and SVM by the following deletions and changes:

- Lower casing text
- Removing usernames
- Removing punctuation (except ‘#’)
- Replacing each URL by ‘URL’
- Replacing each number by ‘0’

Tokenization was done based on whitespace, as tokenization on tweets is non-trivial and wrong tokenization might actually hurt performance.

4.2 Models

Different machine learning models were evaluated and compared, among which Naive Bayes, k-nearest neighbours, RF, SVM, bagging and boosting models, and BiLSTM. Out of these models, the SVM, RF and BiLSTM proved to perform the best. The SVM and RF were implemented using Python’s scikit-learn library (Pedregosa et al., 2011) and the BiLSTM was implemented using Python’s Keras[2] library.

Finally, these models were combined in a majority voting ensemble model. The models are explained in more detail in the next sections, as well as the baseline models.

The architecture of our approach is shown in figure 1.

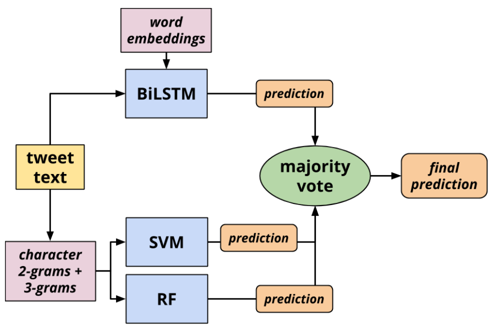


Figure 1: Architecture of our final approach.

[2]http://keras.io/

4.2.1 Baseline

We compared our results to two different baselines provided by the shared task organizers: a linear SVM with default parameters based on a tf-idf representation, and a classifier which assigns the most frequent label in the training set (MFC).

4.2.2 SVM

For the SVM, we tried different types of n-grams as features, including character and word n-grams. In the development phase, the combination of character bi- and trigrams gave the best results. These bi- and trigrams were represented as a tf-idf vector as input for the classifier.

During development, we also fine tuned the hyperparameters of the SVM classifier. The parameter values we changed for the final model were:

- C = 100
- kernel = 'linear'
- probability = True

4.2.3 RF

Just like the SVM classifier, we tried several different features during training for the RF, and again, the character level bi- and trigrams performed the best.

While developing, we fine tuned the parameters of the RF classifier. We experimented with different values and ratios for the number of trees and the maximum depth of the trees. Finally, we found that the following combination of parameter values led to the best performance:

- number of trees = 100
- max tree depth = 49

4.2.4 BiLSTM

In the recent years, neural network algorithms and its variants, proved to give excellent results for many NLP tasks including classification problems. Considering this, we tried a BiLSTM classifier and we used word embeddings taken from van der Goot and van Noord (2017) that were specifically trained on Twitter data. The embeddings were trained with Google's word2vec[3] (Mikolov et al., 2013) tool with 100 dimensions. In the process of training the model, we updated the initial values. The words that were not in the word embeddings were assigned values of 0 for their vector. Moreover, 6,515 words of all 26,026 unique tokens in the dataset (including the test set) were not included in the word embeddings. Among these words one can find unusual hashtags with CamelCasing (e.g. *#SendAllIllegalsHome*), use of repeated emojis and uncanonical usage of words of which some of them can be related to spelling errors.

Finally, we trained the model using a 3 layer BiLSTM model ran with 8 epochs and the following settings:

- 50 hidden units
- batch size = 300
- *adam* optimizer
- dropout rate = 0.2

4.2.5 Ensemble model

The predictions of the RF, SVM and BiLSTM were used in an ensemble model. For each final prediction, the majority vote (MV) of these predictions was taken. This means that the prediction (0 or 1) with the most votes is chosen as final prediction.

In addition to the previously described MV ensemble model, another one was made which uses the confidence scores of the SVM and RF, indicating the chance of being a 0 or 1. Finally, the average of these confidence values and the binary value of the BiLSTM was used to determine the final prediction.

5 Results and Discussion

Table 1 shows the accuracy, precision, recall and F-score for task A (hate speech detection) of the MFC, SVM (baseline), RF, SVM (character n-grams) and BiLSTM separately, as well as the final MV ensemble models and our official results. The official results differ from the other MV because after submission we did small changes in preprocessing and aggregated the development and training data for training the models.

57.3% of the data in each of the datasets (training, development and testing), was annotated as non-hate speech. Therefore, the accuracy of the MFC baseline was 57.3% as well. However, only one of the participants in this shared task managed to beat this baseline.

The accuracies and F-scores were substantially lower on the testing data than on the development data and the scores between the classifiers varied

[3]https://code.google.com/archive/p/word2vec/

Table 1: Evaluation measures (in percentage) of all models for development and testing on task A (hate speech detection). Precision and recall are averaged over the two classes.

	Development				Testing			
Model	Accuracy	Precision	Recall	F-score	Accuracy	Precision	Recall	F-score
MFC baseline*	57.2	32.8	57.3	41.7	**57.9**	28.9	50.0	36.7
SVM baseline*	72.0	71.9	72.1	71.9	49.2	59.5	**54.9**	**45.1**
RF	73.6	73.8	73.6	72.9	42.7	50.0	42.7	29.9
SVM (char. n-grams)	74.0	73.9	74.0	73.9	47.0	63.3	47.0	37.3
BiLSTM	72.2	72.6	72.2	72.3	49.7	65.1	49.7	42.5
Ensemble (MV)	**75.4**	**75.3**	**75.4**	**75.2**	45.6	64.2	45.6	33.7
Ensemble (MV conf. scores)	74.0	74.2	74.0	74.1	48.2	**65.8**	48.2	39.2
Submitted ensemble (MV)	-	-	-	-	46.0	58.1	52.6	39.2

*The baselines for development are re-implemented, so they could be differently compared to the organizer's baseline.

more. This phenomenon can be explained by the fact that the testing data differed a lot from the training data. The organizers stated that before splitting the entire dataset, the data were not shuffled, which can be an explanation for these differences. These differences between the training and test data can lead to overfitting during training and parameter optimization.

During the development phase, character n-grams were the best features for the SVM, but in the testing phase it scored lower than the baseline SVM with a tf-idf representation of word unigrams. Furthermore, the MV ensemble model, combining the binary predictions of the three classifiers, got the highest scores on the development set. As a result, we submitted this MV ensemble model, without confidence scores. In contrast to the development phase, both ensemble models did not perform better on the testing data than the BiLSTM model alone. The BiLSTM performed the best (50% accuracy and 43% F-score) on the test data, but still below the baseline. Furthermore, all models had a higher precision than recall for testing, which also can be attributed to the imbalanced distribution of the data.

Finally, there were some issues with the annotation of the training and testing data. One could find instances of tweets that contain hate speech but were annotated incorrectly in our opinion. Moreover, as it was stated before, only one of the participants outperformed the MFC baseline.

6 Conclusion and future work

For the task of hate speech detection, we incorporated a variety of classifiers (SVM, RF and BiLSTM) and experimented with a range of different features and parameters. At the end we combined the predictions of these models in ensemble models. For development, the SVM, RF and BiLSTM reached similar performances and the ensemble model performed slightly better than these models individually. However, all models scored substantially lower on the test data than on the development data. As a result, we think that our approach led to overfitting on the development set without being able to generalize on the test set.

For future work, there are some ways to improve the results besides experimenting with different classification algorithms. Firstly, Twitter data is noisy and there are many uncanonical words and emojis. We tried to tackle this problem by using word embeddings that were trained on Twitter data for the BiLSTM and character n-grams for the RF and SVM. Another strategy to try is to normalize the data into a more canonical form and feed it to the classifiers. Furthermore, more experiments could be done by incorporating different features and exploiting other information that is available in the data. For example, RF and SVM classifiers trained on word embeddings and the use of certain punctuation marks, emojis and hashtags as separate features could be tried.

Finally, as it was stated in the previous section, we believe there were some issues with the dataset. Moreover, hate speech is hard to define and there is no clear agreement on the definition. This is why we can be skeptical about the annotation procedure. Therefore, we believe better datasets are necessary for the hate speech detection task.

References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Pro-*

ceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019). Association for Computational Linguistics.

Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *ICWSM*, pages 512–515.

Fabio Del Vigna, Andrea Cimino, Felice DellOrletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate me, hate me not: Hate speech detection on facebook. *In Proceedings of the First Italian Conference on Cybersecurity*, pages 86–95.

Rob van der Goot and Gertjan van Noord. 2017. MoNoise: Modeling noise using a modular normalization system. *Computational Linguistics in the Netherlands Journal*, 7:129–144.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Haji Mohammad Saleem, Kelly P. Dillon, Susan Benesch, and Derek Ruths. 2016. A web of hate: Tackling hateful speech in online social spaces. *Proceedings of the 1st Workshop on Text Analytics for Cybersecurity and Online Safety*.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Hajime Watanabe, Mondher Bouazizi, and Tomoaki Ohtsuki. 2018. Hate speech on twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection. *IEEE Access*, 6:13825–13835.

Ziqi Zhang and Lei Luo. 2018. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *CoRR*, abs/1803.03662.

The binary trio at SemEval-2019 Task 5: Multitarget Hate Speech Detection in Tweets

Patricia Chiril
IRIT
Toulouse University
patricia.chiril@irit.fr

Farah Benamara
IRIT, CNRS
Toulouse University
benamara@irit.fr

Véronique Moriceau
LIMSI-CNRS
Univ. Paris-Sud
moriceau@limsi.fr

Abhishek Kumar
Indian Institute of Science
abhishekkumar12@iisc.ac.in

Abstract

The massive growth of user-generated web content through blogs, online forums and most notably, social media networks, led to a large spreading of hatred or abusive messages which have to be moderated. This paper proposes a supervised approach to hate speech detection towards immigrants and women in English tweets. Several models have been developed ranging from feature-engineering approaches to neural ones. We also carried out a detailed error analysis to show main causes of misclassification.

1 Motivation

Social media networks such as Facebook, Twitter, blogs and forums, have become a space where users are free to relate events, personal experiences, but also opinions and sentiments about products, events or other people. This massive growth of user generated web content, along with the interactivity and anonymity the internet provides, may lead to a large spreading of hatred or abusive messages which have to be moderated.

In spite of no universally accepted definition of hate speech and the way it differs from offensive language, there are some common elements that seem to arise. In particular, these messages may express threats, harassment, intimidation or "disparage a person or a group on the basis of some characteristic such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or other characteristic" (Nockleby, 2000).

In this paper, we focus on automatic hate speech detection towards *two different targets* – immigrants and women and we propose several multi-target hate speech detection systems. The task is performed over a collection of English tweets annotated as conveying hate speech against both immigrants and women, as part of HateEval@SemEval2019 (Basile et al., 2019). The first challenge involves building a binary classifier able to determine whether a tweet with a given target (women or immigrants) is hateful or not hateful. For this, we propose both features-based models (relying on both language-dependent and language independent features) and a neural model. We also performed a detailed error analysis to identify main causes of misclassification. Our analysis shows that errors come from several factors, which show the complexity of the task: the presence of irony and sarcasm, the lack of context and implicit hate speech. We also identified tweets for which we question the original label when taking into account the class definition.

The paper is organized as follows. Section 2 briefly presents the current state of the art. Section 3 describes our data and models, while Section 4 analyses the experiments we carried out on multi-target detection. We conclude by providing some perspectives for future work.

2 Related work

Hateful speech can be expressed at different linguistic granularity levels going from lexical to discursive (Cameron, 1992). Both sexism and racism can be expressed explicitly or implicitly (see the following tweets from our data) using different pragmatic devices, including:

- Negative opinion, abusive message: *Stop tweeting about football. You're a girl and you opinion doesn't count. #WomenSuck.*
- Stereotype: *Illegals are dumping their kids heres o they can get welfare, aid and U.S School Ripping off U.S Taxpayers #SendThemBack ! Stop Alowing illegals to Abuse the Taxpayer #Immigration.*
- Humor, irony, sarcasm: *Where is this? Brazil? Uganda? Sudan? Nope, it is France.*

Got to love that cultural enrichment thing going on. #openborders #refugeesnotwelcome #slums.

For most of the harassment and hate speech detection tasks, the classifiers still rely on supervised learning, and when creating a new classifier, one may directly feed different types of features to the classical algorithms (Naive Bayes, Logistic Regression, Random Forest, SVM) or use deep learning methods that will automatically learn abstract features from data instances. Due to the noise present in the data (especially on social media), many authors choose to combine n-grams (due to their high prediction rate) with a large selection of additional features: linguistic features that take into consideration the POS information, dependency relations (long-distance relationship in between words), or word embeddings, which have the advantage of having similar vector representations for different, but semantically similar words. Several approaches incorporate sentiment analysis as a supplementary classification step, assuming that generally negative sentiment relates to a hateful message (Dinakar et al., 2012; Sood et al., 2012).

Although within the Automatic Misogyny Identification shared task at IberEval 2018 the best results were obtained with Support Vector Machine models with different feature configurations, there are also a few notable neural networks techniques deployed in order to detect hate speech in tweets that outperform the existing models: in (Badjatiya et al., 2017) the authors used three methods (Convolutional Neural Network (CNN), Long short-term memory and FastText) combined with either random or GloVe word embeddings. In (Zhang and Luo, 2018) the authors implemented two deep neural network models (CNN + Gated Recurrent Unit layer and CNN + modified CNN layers for feature extraction) in order to classify social media text as racist, sexist, or non-hateful.

3 Multitarget hate speech detection systems

Automatically labelling tweets as hateful or not hateful is a challenging task because the language of tweets is full of grammatically and/or syntactic errors, it lacks conversational context, might consist of only one or a few words and because they can be indirectly hateful (by employing techniques such as sarcasm, satire or irony) it makes the task of text-based feature extraction difficult.

3.1 Data

Our data comes from two corpora. The first one, is an already existing corpus containing English tweets annotated for hate speech against immigrants and women, as part of the HatEval task at SemEval2019 (Basile et al., 2019). The second one was created as a result of the conclusions we had drawn after analyzing the data, i.e. we observed that for most of the tweets, even though the message appeared to be positive, just by having a certain hashtag used, it becomes negative. The hashtag importance is also supported by a simple experiment that includes in the pre-processing step hashtag removal. This leads to a decrease in accuracy by 4% and F-score by 5%. Thus we created a new dataset (DATASET++) by collecting the most used hashtags (we used scrape-twitter[1]) in both hateful (#buildThatWall) and non-hateful tweets (#refugees), as well as the most used hashtags in the misclassified tweets[2].

Table 1 shows the distribution of the tweets for the task of hate speech detection.

Task	#hate	#nonHate	Total
DATASET	5 512	7 559	13 071
DATASET++	17 989	21 921	39 909

Table 1: Tweet distribution in the corpora

For the task at hand, several models have been built, all tested using 10-cross-validation. In the next sections, we detail our models and then provide our results.

3.2 Models

Baseline (**B**)**.** In all the experiments, we used Bag of Words (BoW) model as lexical features. Due to the noise in the data, we performed standard text pre-processing by removing user mentions, URLs, RT, stop words, degraded stop words and the words containing less than 3 characters, and we stemmed all the remaining words by using the Snowball Stemmer[3].

Feature-based models. We experimented with several state of the art features that have shown to

[1]https://www.npmjs.com/package/scrape-twitter

[2]#maga, #usa, #trump, #sendThemBack, #immigration, #noDaca, #deportThemAll, #meToo, #stopTheInvasion, #illegalAliens, #apathyKills, #withImmigrants

[3]http://snowballstem.org

be useful in hate speech detection and we relied on a manually built emoji lexicon that contains 1 644 emojis along with their polarity. We also tested whether by identifying the users opinion we can better classify his attitude as hateful or non-hateful by making use of HurtLex (a multilingual hate word lexicon divided in 17 categories) (Bassignana et al., 2018) and a lexicon containing 1 818 profanity English words created by combining a manually built offensive words list, the noswearing dictionary [4] and an offensive word list[5].

We experimented with several combinations of the features above and we used the best performing ones for training four classifiers:

- C_1 : combines the length of the tweet with the number of words in the HurtLex lexicon with a Baseline architecture
- C_2 : combines the number of words in the offensive lexicon, the number of positive and negative emojis and emoticons and the presence of URLs with a Baseline architecture, but applied on the extended dataset
- C_3 : combines the number of words in the offensive lexicon, the number of positive and negative emojis and emoticons and performs linear dimensionality reduction by means of truncated Singular Value Decomposition and used Random Forest only for intermediate classification, whose output were then combined and passed onto a final Extreme Gradient Booster classifier
- C_4 : the same as C_3 but applied on the extended dataset

Neural model. The last model (C_5) used a Bidirectional LSTM with an attention mechanism. For the task at hand, we used pre-trained on tweets Glove embeddings (Pennington et al., 2014).

4 Results

We tried several machine learning algorithms in order to evaluate and select the best performing one. Hereby, the hate speech system baseline is a Random Forest classifier. Table 2 shows how the experiments were set up and presents the results in terms of accuracy (A), macro-averaged F-score (F), precision (P) and recall (R). For each of the systems we present the results obtained on 10-cross validation (using the provided train, trial and dev datasets) and the official results.

Among the five systems, C_2 represents our best performing one during the development phase [6], while C_5 performed best in the evaluation phase.

Due to a significant decrease in both accuracy and F-score on the official test data, we also investigated the influence of the data distribution in the train and test datasets. The results obtained after shuffling and re-splitting the data (while keeping the original distribution of the tweets) are also presented in Table 2. It is important to realize that these results were obtained by using a train-test configuration on a random test, not by using cross validation. These results are comparable to the ones obtained during the development phase.

As we encountered a significant decrease in the system's performance in the official test, we decided to conduct a deeper analysis in order to identify the main causes of errors.

5 Discussion

Error analysis shows that in the misclassification of hateful instances intervene several factors: the presence of off-topic tweets, the lack of context (as some words that trigger hate in certain contexts may have different connotations in others) and implicit hate speech that employs stereotypes or metaphors in order to convey hatred.

Although the results of the system employed on the extended dataset seemed promising, we couldn't see any improvement on the official test dataset. This might be as a result of not having any information on the actual distribution of the tweets (the number of tweets that convey hate towards immigrants and the number of tweets that convey hate towards women, information that might have been useful when extending the dataset), neither on the way the annotation was done and our definition of hate speech (and the way it differs from offensive language) might have been different. We also identified tweets for which we question the original label when taking into account the class definition. Below, we have provided some examples.

Example 1: The first tweet (annotated as not hateful), containing the users opinion on Poland

[4]https://www.noswearing.com/dictionary

[5]http://www.cs.cmu.edu/ biglou/resources/bad-words.txt

[6]As only the last submitted system was taken into consideration, this also corresponds to the system used for the official ranking

	10-cross validation results				Official results				Train-test configuration on a random test			
	A	F	P	R	A	F	P	R	A	F	P	R
B	0.772	0.762	0.764	0.669	–	–	–	–	0.756	0.747	0.730	0.671
C_1	0.733	0.747	0.623	0.716	0.461	0.401	0.566	0.523	0.757	0.747	0.732	0.666
C_2*	**0.934**	**0.938**	**0.917**	**0.934**	0.437	0.351	0.540	0.508	**0.856**	**0.824**	**0.844**	**0.805**
C_3	0.711	0.707	0.607	0.700	0.450	0.393	0.532	0.513	0.750	0.743	0.701	0.704
C_4	0.924	0.927	0.904	0.923	0.448	0.388	0.530	0.511	0.835	0.820	0.829	0.813
C_5	0.736	0.727	0.709	0.646	**0.5**	**0.471**	**0.581**	**0.551**	0.793	0.779	0.830	0.773

Table 2: Hate speech detection results. Best results are presented in bold font while * marks the system submitted for the official ranking.

receiving immigrants, seems more hateful than the second tweet (hateful), in which the user depicts a series of events.

- If Germans want rapefugees [7] they can keep them. Poland will not accept a single rapefugee. Not even one!!! Poland remains proud and firm! https://t.co/VHoFrxFjB1
- GERAMNY: African rapefugee climbs into house, steel expensive goods, rook a knife and abuse a girl of the family. https://t.co/fVXhWMYcE8

Example 2: Although in the first tweet (not hateful) the user talks about Donald Trump, which doesn't fit in the targeted categories, the annotation raises problems when trying to classify tweets such as the second one (annotated as hateful).

- I love my religious brothers and sisters, but @realDonaldTrump, FUCK YOU, YOU'RE NOT EVEN A REAL THEOCRAT YOU FAT USLESS BITCH.
- @menzemerized_ Worse i have proof. A picture i took of you and one you took of me on the same night. Useless ungreatful kunt!

6 Conclusion

This paper proposed several models that can be used in order to identify messages that convey hate towards women and immigrants, incorporating a variety of features for capturing offensive language. Our results revealed good classication performance on the training dataset, but a lower performance on the evaluation data, with a notable decrease in both accuracy and F-score. Error analysis shows that this decrease is mainly due to the lack of context to infer hateful intents, and the way hate speech was defined in the manual annotation of the dataset.

As the meaning of a message might change in different contexts (as it can be highly dependent on knowledge about the world), in our future work we plan on studying ways to retrieve contextual information.

[7] Accordind to Urban Dictionary, the term rapefugee is usually used when referring to the Muslim refugees coming into Europe in a derogatory way, as refugees are perceived as being more likely to raping people.

Acknowledgments

This work has been funded by Maison des Sciences de l'Homme et de la Société de Toulouse under the project AMeSexTo.

References

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Elisa Bassignana, Valerio Basile, and Viviana Patti. 2018. Hurtlex: A multilingual lexicon of words to hurt. In *5th Italian Conference on Computational Linguistics, CLiC-it 2018*, volume 2253, pages 1–6. CEUR-WS.

Deborah Cameron. 1992. *Feminism and Linguistic Theory.* Palgrave Macmillan.

Karthik Dinakar, Birago Jones, Catherine Havasi, Henry Lieberman, and Rosalind Picard. 2012. Common sense reasoning for detection, prevention, and mitigation of cyberbullying. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2(3):18.

John T. Nockleby. 2000. Hate speech. In *Encyclopedia of the American Constitution (2nd ed., edited by Leonard W. Levy, Kenneth L. Karst et al.*, pages 1277–1279.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Sara Owsley Sood, Elizabeth F Churchill, and Judd Antin. 2012. Automatic identification of personal insults on social news sites. *Journal of the American Society for Information Science and Technology*, 63(2):270–285.

Ziqi Zhang and Lei Luo. 2018. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *arXiv preprint arXiv:1803.03662*.

The Titans at SemEval-2019 Task 5: Detection of hate speech against immigrants and women in Twitter

Avishek Garain
Computer Science and Engineering
Jadavpur University, Kolkata
avishekgarain@gmail.com

Arpan Basu
Computer Science and Engineering
Jadavpur University, Kolkata
arpan0123@gmail.com

Abstract

This system paper is a description of the system submitted to “SemEval-2019 Task 5” Task B for the English language, where we had to primarily detect hate speech and then detect aggressive behaviour and its target audience in Twitter. There were two specific target audiences, immigrants and women. The language of the tweets was English. We were required to first detect whether a tweet is containing hate speech. Thereafter we were required to find whether the tweet was showing aggressive behaviour, and then we had to find whether the targeted audience was an individual or a group of people.

1 Introduction

Hate speech attacks a person or a group on the basis of attributes such as race, religion, ethnic origin, national origin, sex, disability, sexual orientation or gender identity. In the same time, flames (such as rants, taunts, and squalid phrases) are offensive/abusive phrases which might attack or offend the users for a variety of reasons. This is very pertinent due to rise of text messaging through the Internet or cellular phones, which has become a major medium of personal and commercial communication.

Aggression is overt, often harmful, social interaction with the intention of inflicting damage or other unpleasantness upon another individual. It may occur either in retaliation or without provocation. In humans, frustration due to blocked goals can cause aggression. Human aggression can be classified into direct and indirect aggression; whilst the former is characterized by physical or verbal behavior intended to cause harm to someone, the latter is characterized by behavior intended to harm the social relations of an individual or group.

Hate speech and offensive language are pervasive in social media. Online communities, social media platforms, and technology companies have been researching heavily in ways to cope with this phenomena to prevent abusive behavior in social media. This is due to text messaging through the Internet or cellular phones, which has become a major medium of personal and commercial communication.

One of the most effective strategies for tackling this problem is to use computational methods to identify hate speech and aggression in user-generated content (e.g. posts, comments, tweets etc.). This topic has attracted significant attention in recent years of various Natural Language analysts.

The SemEval 2019 task 5 (Basile et al., 2019) was a classification task where we were required to classify a tweet as containing hate speech or otherwise. However, there were some additional challenges presented, which involved automatic detection of aggression, and classification the target audience as an individual or group of people.

To solve the task in hand we built a bidirectional LSTM based neural network for prediction of the three classes present in the provided dataset. In the first subtask our system categorized the instances into `HS` and `NOT`. In the second subtask our system categorized instances into `AGR` and `NOT`. In the third subtask our system categorized instances into `IN` or `GRP`.

The paper has been organized as follows. Section 2 describes a brief survey on the relevant work done in this field. Section 3 describes the data, on which, the task was performed. The methodology followed is described in Section 4. This is followed by the results and concluding remarks in Section 5 and 6 respectively.

2 Related Work

Papers which have been published in the last two years include the surveys by (Schmidt and Wiegand, 2017) and (Fortuna and Nunes, 2018), the paper by (Davidson et al., 2017) presenting the Hate Speech Detection dataset used in (Malmasi and Zampieri, 2017) and a few other recent papers such as (ElSherief et al., 2018; Gambäck and Sikdar, 2017; Zhang et al., 2018).

We were guided by the work of (Zhang et al., 2018) who used a CNN+GRU based approach for a similar task. We use an approach which was influenced by this work but used an LSTM based approach.

A proposal of typology of abusive language sub-tasks is presented in (Waseem et al., 2017). For studies on languages other than English see (Su et al., 2017) on Chinese and (Fišer et al., 2017) on Slovene. Finally, for recent discussion on identifying profanity vs. hate speech see (Malmasi and Zampieri, 2018). This work highlighted the challenges of distinguishing between profanity, and threatening language which may not actually contain profane language.

Previous editions of related workshops are TA-COS[1], Abusive Language Online[2], and TRAC[3] and related shared tasks are GermEval (Wiegand et al., 2018) and TRAC (Kumar et al., 2018).

3 Data

The dataset that was used to train the model is the HatEval dataset (Basile et al., 2019). It was collected from Twitter; the data being retrieved the data using the Twitter API by searching for keywords and constructions that are often included in aggressive messages.

Label	Meaning
HS	Whether the tweet contains hate speech or not
TR	Whether the tweet containing profanity is targeted against some individual/group/others
AG	Whether the tweet contains aggressive behaviour or not

Table 1: Labels used in the dataset

The dataset provided consisted of tweets in their original form along with the corresponding `HS`, `TR` and `AG` labels. The dataset had 9000 instances of training data and 1000 instances of development data. Our approach was to convert the tweet into a sequence of words and then run a neural-network based algorithm on the processed tweet.

Value	HS	TR	AG
0	5217	2442	2224
1	3783	1341	1559
All	9000	3783	3783

Table 2: Distribution of the labels in the training dataset

Value	HS	TR	AG
0	573	208	223
1	427	219	204
All	1000	427	427

Table 3: Distribution of the labels in the development dataset

The provided training and development data were merged and shuffled to create a bigger training set, and we refer to the same as training data when we discuss our methodology.

Value	HS	TR	AG
0	5790	2650	2447
1	4210	1560	1763
All	10000	4210	4210

Table 4: Distribution of the labels in the combined dataset

4 Methodology

The first stage in our pipeline was to preprocess the tweet. This consisted of the following steps:

1. Removing mentions
2. Removing punctuations
3. Removing URLs
4. Contracting whitespace
5. Extracting words from hashtags

The last step (step 5) consists of taking advantage of the Pascal Casing of hashtags (e.g. `#PascalCasing`). A simple regex can extract all words; we ignore a few errors that arise in this procedure. This extraction results in better performance mainly because words in hashtags, to some extent, may convey sentiments of hate. They play an important role during the model-training stage.

We treat the tweet as a sequence of words with interdependence among various words contribut-

[1] `http://ta-cos.org/`

[2] `https://sites.google.com/site/abusivelanguageworkshop2017/`

[3] `https://sites.google.com/view/trac1/home`

ing to its meaning. Hence we use an bidirectional LSTM based approach to capture information from both the past and future context.

Our model is a neural-network based model. First, the input tweet is passed through an embedding layer which transforms the tweet into a 128 length vector. The embedding layer learns the word embeddings from the input tweets. This is followed by two bidirectional LSTM layers containing 64 units each. This is followed by the final output layer of neurons with softmax activation, each neuron predicting a label as present in the dataset. For subtasks 1, 2 and 3, we train separate models containing 2 neurons for predicting `HS(0/1)`, `TR(0/1)` and `AG(0/1)` respectively. Between the LSTM and output layers, we add dropout with a rate of 0.5 as a regularizer. The model is trained using the Adam optimization algorithm with a learning rate of 0.0005 and using crossentropy as the loss.

We note that the dataset is highly skewed in nature. If trained on the entire training dataset without any validation, the model tends to completely overfit to the class with higher frequency as it leads to a higher accuracy score.

To overcome this problem, we took some measures. Firstly, the training data was split into two parts — one for training and one for validation comprising 70 % and 30 % of the dataset respectively. The training was stopped when two consecutive epochs increased the measured loss function value for the validation set.

Secondly, class weights were assigned to the different classes present in the data. The weights were approximately chosen to be proportional to the inverse of the respective frequencies of the classes. Intuitively, the model now gives equal weight to the skewed classes and this penalizes tendencies to overfit to the data.

5 Results

We participated in English Task B of Semeval 2019 task 5 (HatEval) and our system ranks fourth among the competing participants.

We have included the automatically generated tables with our results. We have also included the provided baselines generated by MFC and SVC classifiers respectively. The SVC baseline is generated by a linear SVM based on a TF-IDF representation. The MFC baseline assigns the most frequent label in the training set to all instances present in the test set. We have used these baselines for comparison.

System	Train (%)	Validation (%)
Without	99.82	66.74
With	99.95	70.31

Table 5: Comparison of development phase accuracies with and without hashtag preprocessing

System	F1 (avg)	EMR
MFC baseline	0.421	0.580
SVC baseline	0.578	0.308
BiLSTM	0.471	0.482

Table 6: Overall Metrics

System	F1	Accuracy
MFC baseline	0.367	0.580
SVC baseline	0.45	0.491
BiLSTM	0.484	0.573

Table 7: HS Metrics

System	F1	Accuracy
MFC baseline	0.452	0.824
SVC baseline	0.697	0.785
BiLSTM	0.464	0.817

Table 8: TR Metrics

System	F1	Accuracy
MFC baseline	0.445	0.802
SVC baseline	0.587	0.692
BiLSTM	0.464	0.763

Table 9: AG Metrics

6 Conclusion

Here we have presented a model which performs satisfactorily in the given tasks. The model is based on a simple architecture. There is scope for improvement by including more features (like those removed in the preprocessing step) to increase performance. Another drawback of the model is that it does not use any external data other than the dataset provided which may lead to poor results based on the modest size of the data. Related domain knowledge may be exploited to obtain better results.

References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Darja Fišer, Tomaž Erjavec, and Nikola Ljubešić. 2017. Legal Framework, Dataset and Annotation Schema for Socially Unacceptable On-line Discourse Practices in Slovene. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hatespeech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 467–472.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Huei-Po Su, Chen-Jie Huang, Hao-Tsung Chang, and Chuan-Jie Lin. 2017. Rephrasing Profanity in Chinese Text. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

TuEval at SemEval-2019 Task 5: LSTM Approach to Hate Speech Detection in English and Spanish

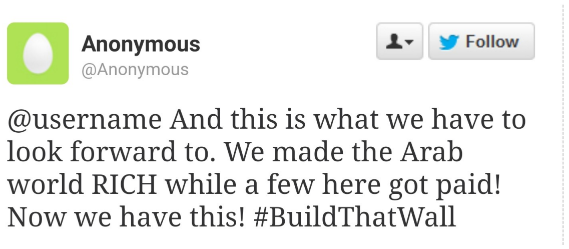
Mihai Manolescu **Denise Löfflad** **Adham Nasser Mohamed Saber**
Masoumeh Moradipour Tari
{mihai.manolescu, denise.loefflad, adham-nasser.mohamed-saber, masoumeh.moradipour-tari}@student.uni-tuebingen.de
Eberhard-Karls University of Tübingen

Abstract

The detection of hate speech, especially in online platforms and forums, is quickly becoming a hot topic as anti-hate speech legislation begins to be applied to public discourse online. The HatEval shared task was created with this in mind; participants were expected to develop a model capable of determining whether or not input (in this case, Twitter posts in English and Spanish) could be considered hate speech (designated as Subtask A), if they were aggressive, and whether the tweet was targeting an individual, or speaking generally (Subtask B). We approached this Subtask by creating a LSTM model with an embedding layer. We found that our model performed considerably better on English language input when compared to Spanish language input. In English, we achieved an F1-Score of 0.466 for Subtask A and 0.462 for Subtask B; In Spanish, we achieved scores of 0.617 and 0.612 on Subtask A and Subtask B, respectively.

1 Introduction

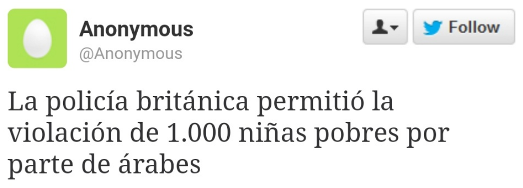
Social media plays an important role nowadays and dominates everyday life. Social networks like Facebook, Twitter and Instagram are platforms where people express thoughts, feelings and emotions regarding themselves or others. This can lead to different opinions colliding and creating conflicts. Often, feelings are not expressed objectively and can be offensive to other users. In order to make social media more comfortable, so called hate speech needs to be detected and removed. Hate speech is here defined as: Any communication that disparages a person or a group on the basis of some characteristic such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics (Basile et al., 2019). To assure there is no spread of illegal hate speech, the EU has created a code of conduct for social media platforms (European Union, 2018) that needs to be followed. According to these EU regulations, social media platforms must regulate hateful speech. In addition, social media occupies an increasingly larger portion of public discourse; even without these EU regulations, it seems that these platforms should have some methods for controlling violent discourse.
For these reasons, the HatEval shared task (Basile et al., 2019) was created. The task is divided into two subtasks; Subtask A is hate speech detection against immigrants and women, a binary classification problem where a tweet is classified as either hateful or not hateful. Subtask B is determining whether a given tweet is aggressive, and whether it is targeting an individual, or not referring to any particular person. Further, each of these Subtasks is evaluated on English tweets and using Spanish tweets. We were provided a 9000-tweet English training set, and a 5000 tweet Spanish training set. The training sets were manually tagged as hateful or not hateful, aggressive or not aggressive, and targeted or not targeted - examples of tweets marked as hateful can be seen in Figures 1 and 2 below.



@username And this is what we have to look forward to. We made the Arab world RICH while a few here got paid! Now we have this! #BuildThatWall

Figure 1: Example of a hateful English tweet.

In this paper, we detail our methods for approaching these problems. We will first cover related works before detailing our specific solutions for Subtask A and Subtask B; we will then cover our model (and previously attempted


Anonymous
@Anonymous
Follow

La policía británica permitió la violación de 1.000 niñas pobres por parte de árabes

Figure 2: Example of a hateful Spanish tweet.

models) and present our results. Hate speech detection is naturally a far reaching topic, and in conclusion we will discuss the implications of our work for the field in general.

2 Related Work

To begin, we attempted to take a brief survey of previous work in the field of hate speech detection. Since this is, at heart, a binary classification task, we saw that there were many established approaches to solving this problem - various machine learning techniques, according to our research, were shown to be valid, such as Recurrent and Convolutional Neural Networks (RNN and CNNs) (Stammbach et al., 2018), Support Vector Machines (SVMs) (Malmasi and Zampieri, 2017), Long Short Term Memory models (LSTMs) (Zhang et al., 2015; Risch et al., 2018), as well as simpler linear regression approaches (Kent, 2018). In our estimation, we determined that LSTM approaches were most successful (Golem et al., 2018; Del Vigna12 et al., 2017), and took such an approach in the creation of our model. Some other approaches were too computationally expensive; in addition, we felt that, due to the nebulous nature of hate speech determination, the additional information captured by an LSTM model would be worthwhile in these tasks. We also determined that, for such a task, the use of non-word features would be superfluous, as previous work had shown them to decrease performance (Stammbach et al., 2018), and this was supported by other works on simple classification tasks, even when LSTMs or RNNs were not used (Malmasi and Zampieri, 2017). Research showed that various features, including emoticons, sentiment analysis, and number of characters tended to hurt performance (Kent, 2018).
Predictably, most work done on this topic has focused on English language data; we found only a few papers on Spanish language hate speech detection (Álvarez-Carmona et al., 2018; Fersini et al., 2018), which we attempted to use to ensure our model would function across language boundaries.

3 Model

At the outset, we employed a simple unidirectional, 1-layer LSTM model. As we saw preliminary results we altered our model accordingly. We also attempted to use a 2-layer LSTM model, and settled on a 1-layer LSTM model with a simple embedding layer, using mainly the *Keras* (Chollet et al., 2015) library.

3.1 Pre-Processing

Based on our research, we saw that limited pre-processing of the data set could improve performance; to that end, the following pre-processing steps were taken:

- replace usernames with username markers
- remove punctuation and special characters (@ / , ; . : ? ¿ ¡ ! $)
- lowercase

We made the decision not to omit hashtags; while usernames do not necessarily convey information pertinent to the tweet itself, it was determined that hashtags are frequently used for meaningful purposes and must be considered when attempting to classify Twitter data. We attempted to expand our pre-processing efforts when dealing with Spanish language data after seeing early results (replacing characters such as ‘ñ´ with ‘n´ for example), but without success; such efforts hurt our model more than they helped.

3.2 Recurrent Neural Network

Our model used character based representations of all data. We used an embedding layer with input dimension of 5000 and an output of 28; input length was determined by finding the length of the longest item in the data set, and padding all representations to this length. Additionally, we used an LSTM layer with 64 units, with a dropout rate of 0.1 (determined after simple trial and error tests), and our model employed a sigmoid activation function and a binary cross entropy loss function. Our model was trained for 50 epochs on the English language dataset, and 20 epochs on the Spanish language dataset.

Models	F1-score
1-Layer LSTM	0.31
2-Layer LSTM	0.42
1-Layer LSTM w/ Embedding	0.69

Table 1: Development set F1-scores for preliminary testing of models.

4 Evaluation

We first evaluated our preliminary models using the development data set, specifically using results of Subtask A in English to determine which of our beginning approaches was most successful. After this determination, we expanded upon our best working model (the simple LSTM model with embedding layer), and proceeded to use this approach to handle all tasks in both languages. We calculated the F1-score for each of our models, and used this for our evaluations. As shown in Table 1, the 1-Layer LSTM model with Embedding outperformed our other two models significantly and achieved an F1-Score of 0.69 on the development set.

tasks	Accuracy	Precision	Recall	F1-score	EMR
Subtask A (En.)	0.488	0.548	0.533	0.466	N/A
Subtask B (En.)	0.565	0.497	0.482	0.462	0.173
Subtask A (Sp.)	0.630	0.618	0.617	0.617	N/A
Subtask B (Sp.)	0.680	0.629	0.608	0.612	0.428

Table 2: Results for Tasks A and B in English and Spanish.

The average results for each metric are shown in Table 2. The final ranking for Subtask A for English, as well as Spanish, was based on the F1-score. Our F1-score was 0.466 for English, which ranked us 27th out of 69 teams that submitted a result for this Task. Since we had some problems with the Spanish data set, we could only submit one solution for Subtask A, which placed us 36th out of 39 teams.

Evaluation for Subtask B was based on two criteria - partial match and exact match. For partial match, each dimension that needs to be predicted, is being looked at independently and therefore the usual evaluation metrics are being used (Precision, Accuracy, Recall and F1-Score). For the exact match all the dimensions to be predicted are jointly considered. Ranking was solely based on the score of the *Exact Match Ratio* (EMR). For English we achieved an EMR score of 0.173, which ranked us second to last, even if our average F1-score was higher than other systems'. Since we had the same problems as in Subtask A, we again were only able to submit one file in Subtask B for Spanish, where we achieved an EMR of 0.428 and an average F1-Score of 0.612. The significant difference for the F1-Score between English and Spanish comes from the fact that there was less training data for Spanish compared to English.

5 Conclusion & Future Work

We created a simple LSTM model and applied it to all tasks - detecting hate speech, determining aggression, and determining targeted or general speech, achieving F1-scores of 0.466 and 0.462 for Subtask A and B in English, and scores of 0.617 and 0.612 for Tasks A and B in Spanish. In our work, we saw that our model performed considerably better on English language data when compared to Spanish language data. We were not able to reduce this discrepancy with additional pre-processing of Spanish language data. The difference in performance may be explained by the nature of Spanish language discourse online - perhaps there is greater accent- or dialect-based difference in Spanish when compared to English (Çöltekin and Rama, 2018), which could confound attempts to train a model off of a Spanish language corpus that does not specifically control for dialect.

There is still much work to be done in the field of hate speech evaluation. It is possible that a large improvement in performance would be seen if word representations were used instead of character representations; much of the vocabulary of online communication and discourse involves the use of colloquialisms, informal speech, and metaphorical language, which word based representations could perhaps better capture. Further, contextual information from the rest of a particular tweet could also help in determining whether or not a given word is being used in a malicious way; this information could have been captured through the use of n-gram models or contextual word representation methods. Using meta-information about a particular user, topic, or hashtag could have also improve performance (Schmidt and Wiegand, 2017); such methods go outside the scope of the shared task, but it is conceivable that a platform such as Twitter could consider previous tweets of a given user, or perhaps topic modelling methods, in a commercial hate speech detection model (for example, it seems rational to consider a tweet with a topic such as 'right wing politics' more likely to be hate speech than a tweet with the topic 'gardening'). The use of lexical resources like lists of slurs have also shown to be effective in combination with other features (Schmidt and Wiegand, 2017; Davidson et al., 2017). Work could also be done in hate speech detection in long form documents; it goes without saying that a model that can effectively detect hate speech in short, one- to three-sentence tweets will not necessarily perform as well on longer corpora, such as articles. In these cases, context-based word representations, n-gram models, etc. could become even more valuable.

References

Álvarez-Carmona, M. Á., Guzmán-Falcón, E., Montes-y Gómez, M., Escalante, H. J., Villasenor-Pineda, L., Reyes-Meza, V., and Rico-Sulayes, A. (2018). Overview of MEX-A3T at IberEval 2018: Authorship and Aggressiveness Analysis in Mexican Spanish Tweets. In *Notebook Papers of 3rd SEPLN Workshop on Evaluation of Human Language Technologies for Iberian Languages (IBEREVAL), Seville, Spain*.

Basile, V., Bosco, C., Fersini, E., Nozza, D., Patti, V., Rangel, F., Rosso, P., and Sanguinetti, M. (2019). Semeval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Chollet, F. et al. (2015). Keras. https://keras.io.

Çöltekin, Ç. and Rama, T. (2018). Tübingen-Oslo at Semeval-2018 task 2: SVMs perform better than RNNs in Emoji Prediction. In *Proceedings of The 12th International Workshop on Semantic Evaluation*.

Davidson, T., Warmsley, D., Macy, M., and Weber, I. (2017). Automated Hate Speech Detection and the Problem of offensive language. In *Eleventh International AAAI Conference on Web and Social Media*.

Del Vigna12, F., Cimino23, A., Dell'Orletta, F., Petrocchi, M., and Tesconi, M. (2017). Hate me, hate me not: Hate speech detection on Facebook.

European Union (2018). Code of Conduct on countering illegal Hate Speech online. https://ec.europa.eu/info/sites/info/files/code_of_conduct_on_countering_illegal_hate_speech_online_en.pdf. [Online; accessed 16-February-2019].

Fersini, E., Rosso, P., and Anzovino, M. (2018). Overview of the Task on Automatic Misogyny Identification at Ibereval 2018. *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018), co-located with 34th Conference of the Spanish Society for Natural Language Processing (SEPLN 2018). CEUR Workshop Proceedings. CEUR-WS.org*.

Golem, V., Karan, M., and Šnajder, J. (2018). Combining Shallow and Deep Learning for Aggressive Text Detection. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*.

Kent, S. (2018). German Hate Speech Detection on Twitter. *Proceedings of the GermEval 2018 Workshop*.

Malmasi, S. and Zampieri, M. (2017). Detecting Hate Speech in Social Media. *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*.

Risch, J., Krebs, E., Löser, A., Riese, A., and Krestel, R. (2018). Fine-Grained Classification of Offensive Language. *Proceedings of the GermEval 2018 Workshop*.

Schmidt, A. and Wiegand, M. (2017). A Survey on Hate Speech Detection using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*.

Stammbach, D., Zahraei, A., Stadnikova, P., and Klakow, D. (2018). Offensive Language Detection with Neural Networks for Germeval Task 2018. *Proceedings of the GermEval 2018 Workshop*.

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level Convolutional Networks for Text Classification. In *Advances in neural information processing systems*.

Tw-StAR at SemEval-2019 Task 5: N-gram embeddings for Hate Speech Detection in Multilingual Tweets

Hala Mulki[*], **Chedi Bechikh Ali**[**], **Hatem Haddad**[†§] **and Ismail Babaoğlu**[*]

[*]Department of Computer Engineering, Selcuk University, Turkey
[**]LISI Laboratory, INSAT, Carthage University, Tunisia
[†]RIADI Laboratory, National School of Computer Sciences, University of Manouba, Tunisia
[§]iCompass Consulting, Tunisia
halamulki@selcuk.edu.tr,chedi.bechikh@gmail.com
haddad.Hatem@gmail.com,ibabaoglu@selcuk.edu.tr

Abstract

In this paper, we describe our contribution in SemEval-2019: subtask A of task 5 "Multilingual detection of hate speech against immigrants and women in Twitter (HatEval)". We developed two hate speech detection model variants through Tw-StAR framework. While the first model adopted one-hot encoding n-grams to train an NB classifier, the second generated and learned n-gram embeddings within a feedforward neural network. For both models, specific terms, selected via MWT patterns, were tagged in the input data. With two feature types employed, we could investigate the ability of n-gram embeddings to rival one-hot n-grams. Our results showed that in English, n-gram embeddings outperformed one-hot n-grams. However, representing Spanish tweets by one-hot n-grams yielded a slightly better performance compared to that of n-gram embeddings. The official ranking indicated that Tw-StAR ranked 9th for English and 20th for Spanish.

1 Introduction

Under the guise of free speech, social media systems have been misused by some users who embed hatred, offensive, racist or negative stereotyping contents within their shared posts. Unfortunately, online Hate Speech (HS) is spreading widely, forming a serious problem that can lead to actual hate crimes (Matsuda, 2018). Many countries adopted laws prohibiting HS where people convicted of using HS can face large fines and even imprisonment. Although Twitter has its anti HS policy[*], the increasing size of the daily-shared tweets in addition to multilingualism and informal writing issues evoke the necessity for automatic HS detection in tweets.

Hate speech detection problem has been addressed as a machine learning classification task. Recent studies proposed multiple HS detection models with different characteristic in terms of features, classification algorithms and implementation architectures. While some HS models employed hand-crafted features generated by NLP tools and external semantic resources, other models adopted text embedding features that are automatically learned from the corpus itself. Both feature types were fed to train either traditional classifiers such as Support Vector Machines (SVM), Naive Bayes (NB) and so forth, or more complicated deep learning-based classifiers such as Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM) and Recurrent Neural Network (RNN) (Schmidt and Wiegand, 2017). The variety of hand-crafted features enabled obtaining reliable performances. However, generating such features based on morphological NLP tools or semantic resources remains laborious. In contrast, embedding features are easier to generate and can yield good HS classification results when used within deep learning architectures (Yuan et al., 2016). Nevertheless, producing good performances via deep neural systems requires providing large-sized labeled training data, tuning many hyper parameters and high computation/time cost. In line with Tw-StAR framework (Mulki et al., 2017, 2018a), we propose, here, an HS model based on the hypothesis that, pairing between n-gram embeddings and less-complicated architectures i.e. feedforward neural network can lead to an efficient HS detection with least complexity.

2 Hate Speech Detection Models

According to the used features, HS detection models can be classified into hand-crafted-based and text embeddings-based.

[*]support.twitter.com/articles/20175050

2.1 Hand-Crafted-based Models

Being a user-generated content, HS terms tend to have variant writing shapes. (Waseem and Hovy, 2016) handled this issue by using char-grams to train an LR classifier. Combining char-grams with extra linguistic features such as word n-grams and user's gender improved the performance.

Additional user-related features were studied in (Unsvåg and Gambäck, 2018) within a multilingual HS detection task. Single and combined features were fed into an LR classifier. The study showed that specific user features favorably impact the performance.

The winning system (Pamungkas et al., 2018) in misogyny detection contest (Fersini et al., 2018) examined several sets of hand-crafted features including stylistic, lexical and structural. The features were formulated within one-hot/sparse encoding vectors and fed into an SVM classifier. It was noted that using features from HurtLex lexicon (Bassignana et al., 2018) enriched the lexical features set and enhanced the performance.

2.2 Text Embeddings-based Models

In these models, the input text is represented using dense, low-dimentional and real-valued vectors. In (Nobata et al., 2016), a comprehensive comparison was conducted among three embedding feature types: doc2vec (Le and Mikolov, 2014), word2vec and pretrained word embeddings against hand-crafted features. Using a regression model trained with the previous features, it could be noted that while doc2vec embeddings outperformed the other embedding features, combining them with all other features could further enhance the HS content recognition.

(Badjatiya et al., 2017) explored CNN, LSTM and FastText models to learn embedding features needed to classify HS contents. These models were trained by embedding features and evaluated against each other and towards SVM, LR and GBDT classifiers trained with hand-crafted features. Moreover, the authors explored training an GDBT classifier with word embeddings learned via various deep models. While CNN was the best-performing deep model, using the word embeddings learned via LSTM to train the simple less-complicated GDBT classifier improved the results.

In (Gambäck and Sikdar, 2017), context-aware word embeddings learned by word2vec, char 4-grams and a combination of both were used to train a CNN-based classifier. The proposed model was compared with an LR classifier trained via n-gram features (Waseem and Hovy, 2016). The results showed that regardless of the used embeddings type, CNN model outperformed the baseline model. Moreover, word2vec embeddings were of the best classification performance among the other embedding features.

3 Tw-StAR HS Detection Model

To detect HS in English and Spanish datasets provided by (Basile et al., 2019), Tw-StAR (see Figure 1) was applied through the following steps:

3.1 Preprocessing

- Initial preprocessing: includes removing the non-sentimental content such as URLs, usernames, digits, hashtag symbols and punctuation from both datasets (Mulki et al., 2018b).

- Stopwords removal: for English and Spanish, we removed stopwords using 1,012 English stopwords and 731 Spanish stopwrods derived from Terrier package[†] and snowball[‡], respectively.

- Lemmatization: we adopted Treetagger lemmatizer (Schmid, 1999); as it was used successfully for English and Spanish in (Mulki et al., 2018a). TreeTagger forms a language-independent tool to annotate texts with part-of-speech and lemma information.

- Hate indicatives tagging: Multi-word terms (MWT) are meaning indicators of a sentence/document (Henry et al., 2018; Bechikh-Ali et al., 2019). In our case, they can represent the entities discussed within a tweet. As our objective is to infer HS in tweets, we believe that recognizing MWT can assist in identifying the important entities related to hate speech or victims of hate speech. This has been practically noticed among the MWT extracted from the training set as we can mention: african_migrant, Iraqi_refugee_terrorist, Muslim_refugee, immigration_negative_effect. It should be noted that, MWT were extracted from hate tweets contained in the training set. Later, the extracted MWT were replaced in both training

[†]https://bitbucket.org/kganes2/
[‡]http://snowball.tartarus.org/

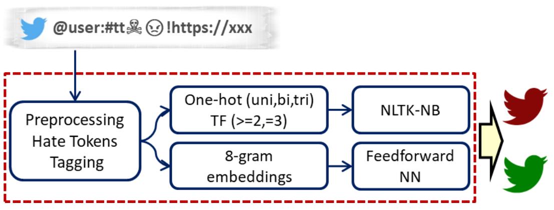

@user:#tt !https://xxx
Preprocessing Hate Tokens Tagging
One-hot (uni,bi,tri) TF (>=2,=3)
8-gram embeddings
NLTK-NB
Feedforward NN

Figure 1: Tw-StAR framework

and dev/test sets with the tag “HateWord”. MWT identification process was performed through two steps: (a) Shallow syntactic parsing where each word was tagged with its syntactic category using Treetagger that supports English and Spanish, and (b) MWT extraction conducted based on specific syntactic patterns of noun and adjective combinations using this schema:

$$MWT=(Adj|N)^*(N|NP)(N|Adj|NP)^*$$

where * denotes a list of 0 or more elements, the MWT length varies between 2 and 4 words. Adj, N and NP refer to adjective, noun and proper noun, respectively.

3.2 Feature Extraction

Two types of features were generated to train both model variants of Tw-StAR.

- One-hot n-grams: are generated by subjecting the preprocessed tweets to tokenization. Three N-grams schemes including unigrams, bigrams and trigrams were adopted. For a certain n-grams scheme, a tweet’s feature vector is constructed via examining the presence/absence of this scheme among the tweet’s tokens. Thus, the feature vectors are formulated as one-hot encoding vectors with binary values “1” (presence) or “0” (absence). Term frequency (TF) property was employed to reduce the features size according to predefined frequency thresholds.

- N-gram embeddings: Based on word embeddings initialized randomly at the embedding layer of Tw-StAR Feedforward neural model, n-gram embeddings are produced by applying a composition function over a specific number of word embedding vectors. In our experiments, we used the additive composition function, known as Sum Of Word Embeddings (SOWE). While composing an n-gram embedding vector, by performing an element-wise sum over word embedding vectors, SOWE considers the co-occurrence information of the n-gram words and totally ignores the local word order.

3.3 Hate Speech Classification

Using the generated features a Naive Bayes (NB) classifier and a feedforward neural network model were trained:

- Naive Bayes model: with one-hot n-gram features, we used an NB classifier implemented as a multinomial NB decision rule together with binary-valued features.

- Feedforward neural network : this model was developed with the following layers:
  - Embeddings layer receives the n-grams generated for each input tweet and map their constituent words into their corresponding word dense representations. N-grams are produced by going through the tweet using a sliding window of a fixed size (N) such that each word of the tweet is considered. All the resulting n-grams (shingles) are then fed to the model with supervision information included where each n-gram is associated with 2-dimension labels HS [1,0] or NOT [0,1] that represent the polarity of the tweet from which the n-gram is derived.
  - Lambda layer composes n-gram embeddings by applying SOWE over the word embeddings resulting from the embedding layer.
  - Hidden layer introduces non-linear discriminating features to the model with Relu activation function.
  - Output layer is equipped with a softmax function to induce the estimated probabilities of each n-gram output label (HS/NOT). Considering the whole tweet, HS scores and NOT scores predicted for all n-grams of the tweet are summed, then each of which is divided by the number of n-grams, contained in a tweet, yielding two values for the potential HS and NOT scores of the tweet. The label of the tweet is, thus, decided according to the greater among these two values.

Lang.	Features	R.	F1	Acc.
English	uni+bi	0.85	0.87	0.89
	8-gram emb.	**0.98**	**0.94**	**0.95**
Spanish	uni+bi	**0.77**	**0.77**	**0.78**
	8-gram emb.	0.72	0.72	0.72

Table 1: Unigrams+bigrams (TF threshold=2) and 8-gram embeddings results of NB/neural models for train/dev sets.

L.	Team (F1 rank)	P.	R.	F1	Acc.
Eng.	saradhix (1)	**0.69**	**0.68**	**0.65**	**0.65**
	Panaetius (2)	0.59	0.59	0.57	0.57
	YunxiaDing (3)	0.64	0.603	0.55	0.56
	Tw-StAR (9)	0.54	0.53	0.5	0.54
Sp.	luiso.vega (1)	0.73	0.74	**0.73**	**0.73**
	francolq2 (2)	0.73	0.74	**0.73**	**0.73**
	gertner (3)	**0.75**	**0.75**	**0.73**	**0.73**
	Tw-StAR (20)	0.70	0.71	0.70	0.70

Table 2: Tw-StAR official Codalab ranking.

4 Results and Discussion

Having the data preprocessed and hate indicatives specified and tagged in both training and dev/test sets, two HS models were used.

The first model is an NB classifier from NLTK[§] trained with one-hot n-gram features. We generated three n-gram schemes: unigrams (uni), unigrams+bigrams (uni+bi) and unigrams+bigrams+trigrams (uni+bi+tri). NB was first trained using all n-gram features, then by a reduced number of features obtained via term frequency (TF) with two threshold: 2 and 3. Among several runs with various n-gram schemes and TF values, we adopted the best-performing scheme: uni+bi and TF threshold: 2.

The second model combines n-gram embeddings within a feedforward neural network. The window size 8 was, empirically, selected to produce 8-gram embeddings. Similarly, the embeddings dimension value was set to 100. For training, backpropagation algorithm and "Adam" optimizer (Kingma and Ba, 2014) were used.

Table 1 lists the results obtained using Train and Dev sets of English and Spanish tweets where the language, embeddings, average recall, average f-measure and accuracy are referred to as (Lang.), (emb.), (R.), (F1) and (Acc.), respectively.

Considering Table 1, both feature types performed well for HS detection in English. However, n-gram embeddings were better with an F1 of 94% against 87% scored by one-hot n-grams. We can explain that by the ability of n-gram embeddings to capture the semantic word regularities regardless of the local word order; which is appropriate to handle the informal English used on Twitter; where varying word orders can infer the same semantics (Iyyer et al., 2015).

Regarding the Spanish dataset, while the HS classification performances produced by both feature types were quite comparable, one-hot n-grams achieved slightly better results with an F1 77% and accuracy of 78% compared to 72% and 72% scored by n-gram embeddings, respectively. This could be attributed to the differences in vocabulary introduced by the different spoken varieties of Spanish found in the tweets (Maier and Gómez-Rodríguez, 2014). Hence, SOWE may miss the synonymous and semantic relations among such different words having same/close semantics which, in turn, leads to less expressive n-gram embeddings.

Having the best-performing features identified for English and Spanish, we adopted one-hot n-grams for Spanish and n-gram embeddings for English in the official submission. Table 2 lists the official results of Tw-StAR against the top three ranking systems where (L.), (Acc.), (Eng.), (Sp.), (R.) and (F1) refer to language, accuracy, English, Spanish, recall and f-measure, respectively.

Considering Table 1 and Table 2, we observe that Tw-StAR exhibit a robust performance for the Spanish dataset, while the evaluation measures degraded for the English dataset. We believe that, this could be attributed to the lack of homogeneity between the train/dev and test sets of English data.

5 Conclusion

We developed two HS detection models for multilingual tweets. With two feature types used, we investigated how likely n-gram embeddings can rival one-hot n-grams in HS detection. Upon training NB and a feedforward neural net with one-hot n-grams and n-gram embeddings, respectively, n-gram embeddings exhibited a better performance in English while the vocabulary differences in Spanish made n-gram embeddings less expressive. For future work, we aim to target HS in underrepresented languages such as Arabic and Turkish.

[§]https://www.nltk.org

References

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Elisa Bassignana, Valerio Basile, and Viviana Patti. 2018. Hurtlex: A multilingual lexicon of words to hurt. In *5th Italian Conference on Computational Linguistics, CLiC-it 2018*, volume 2253, pages 1–6. CEUR-WS.

Chedi Bechikh-Ali, Hatem Haddad, and Yahya Slimani. 2019. Empirical evaluation of compounds indexing for turkish texts. *Computer Speech and Language*, 56(1):95–106.

Elisabetta Fersini, Paolo Rosso, and Maria Anzovino. 2018. Overview of the task on automatic misogyny identification at ibereval 2018.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using convolutional neural networks to classify hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Sam Henry, Clint Cuffy, and Bridget T. McInnes. 2018. Vector representations of multi-word terms for semantic relatedness. *Journal of Biomedical Informatics*, 77:111 – 119.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1681–1691.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Wolfgang Maier and Carlos Gómez-Rodríguez. 2014. Language variety identification in spanish tweets. In *Proceedings of the EMNLP'2014 Workshop on Language Technology for Closely Related Languages and Language Variants*, pages 25–35.

Mari J Matsuda. 2018. Public response to racist speech: Considering the victim's story. In *Words that wound*, pages 17–51. Routledge.

Hala Mulki, Chedi Bechikh Ali, Hatem Haddad, and Ismail Babaoglu. 2018a. Tw-star at semeval-2018 task 1: Preprocessing impact on multi-label emotion classification. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 167–171.

Hala Mulki, Hatem Haddad, Chedi Bechikh Ali, and Ismail Babaoğlu. 2018b. Tunisian dialect sentiment analysis: A natural language processing-based approach. *Computación y Sistemas*, 22(4).

Hala Mulki, Hatem Haddad, Mourad Gridach, and Ismail Babaoğlu. 2017. Tw-star at semeval-2017 task 4: Sentiment classification of arabic tweets. In *Proceedings of the 11th international workshop on semantic evaluation (SEMEVAL-2017)*, pages 664–669.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153. International World Wide Web Conferences Steering Committee.

Endang Wahyu Pamungkas, Alessandra Teresa Cignarella, Valerio Basile, Viviana Patti, et al. 2018. 14-exlab@ unito for ami at ibereval2018: Exploiting lexical knowledge for detecting misogyny in english and spanish tweets. In *3rd Workshop on Evaluation of Human Language Technologies for Iberian Languages, IberEval 2018*, volume 2150, pages 234–241. CEUR-WS.

Helmut Schmid. 1999. Improvements in part-of-speech tagging with an application to german. In *Natural language processing using very large corpora*, pages 13–25. Springer.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Elise Fehn Unsvåg and Björn Gambäck. 2018. The effects of user features on twitter hate speech detection. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 75–85.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Shuhan Yuan, Xintao Wu, and Yang Xiang. 2016. A two phase deep learning model for identifying discrimination from tweets. In *EDBT*, pages 696–697.

UA at SemEval-2019 Task 5: Setting a Strong Linear Baseline for Hate Speech Detection

Carlos Perelló$^\diamondsuit$ **David Tomás**$^\diamondsuit$ **Alberto Garcia-Garcia**$^\diamondsuit$
Jose Garcia-Rodriguez$^\diamondsuit$ **Jose Camacho-Collados**$^\clubsuit$
$^\diamondsuit$ University of Alicante, Spain
$^\clubsuit$ Cardiff University, United Kingdom
$^\diamondsuit$cpc69@alu.ua.es,dtomas@dlsi.ua.es,{agarcia,jgarcia}@dtic.ua.es,
$^\clubsuit$camachocolladosj@cardiff.ac.uk

Abstract

This paper describes the system developed at the University of Alicante (UA) for the SemEval 2019 Task 5: Multilingual detection of hate speech against immigrants and women in Twitter. The purpose of this work is to build a strong baseline for hate speech detection by means of a traditional machine learning approach with standard textual features, which could serve as a reference to compare with deep learning systems. We participated in both task A (Hate Speech Detection against Immigrants and Women) and task B (Aggressive behavior and Target Classification) for both English and Spanish. Given the text of a tweet, task A consists of detecting hate speech against women or immigrants in the text, whereas task B consists of identifying the target harassed as individual or generic, and to classify hateful tweets as aggressive or not aggressive. Despite its simplicity, our system obtained a remarkable macro-F1 score of 72.5 (sixth highest) and an accuracy of 73.6 (second highest) in Spanish (task A), outperforming more complex neural models from a total of 40 participant systems.

1 Introduction

Due to the massive rise of users in social media, the presence of verbal abuse, hate speech and bully-attitudes has increased over the years. A clear example is Twitter, where users find ways to anonymously harass and offend other individuals or collectives. This is especially troublesome as hate speech and hate crime are strongly related. Therefore, an early detection of hate speech could help prevent the subsequent hate crime. Online platforms like Twitter have been seeking to combat hate speech on their site, but it still requires a lot of manual work because there is not a reliable automatic method that can correctly identify hate speech behaviour. Building such automatic (or semi-automatic) systems is therefore essential to effectively fight this problem.

Hate speech detection is still a challenging task due to a number of reasons. First, hate speech content tends to be ambiguous and context-dependant (Chatzakou et al., 2017). Moreover, hate speech can cross sentence boundaries and be present in sarcastic comments in the same voice as the people that were producing abusive languages. These and other issues for detecting hate speech are discussed in more detail in Nobata et al. (2016).

In order to deal with these issues, over the past few years several techniques to detect hate speech and abusive language online have been proposed.[1] Previous works made use of heterogeneous features such as bag of words, n-grams, punctuation, as well as lexical features and user-related features (Chatzakou et al., 2017). In addition to these features, previous approaches showed the effectiveness of using word embeddings to detect abusive language in social media (Djuric et al., 2015) and exposed how sentiment analysis can also contribute to hate speech and offensive language detection (Nahar et al., 2012).

In this paper we build on these earlier works and propose a comprehensive framework to develop a traditional machine learning-based approach to hate speech detection, with the purpose of serving as a strong baseline for future systems using deep learning techniques. Our framework will be based on a linear classifier with standard textual features. As we will show throughout the paper, n-grams provide a reliable starting point when facing hate speech classification, and the performance can be further improved when combined with word em-

[1]Although related, it is important to distinguish between hate speech and abusive or offensive language. While the former is used to express hatred towards a targeted group based on characteristics such as race, ethnicity, gender, and sexual orientation, the latter can be used in the usual language of some users without being hateful (Davidson et al., 2017).

beddings and sentiment analysis features.

In particular for this work, we focus on hate speech against women and immigrants, following the Task 5 of SemEval 2019 (Basile et al., 2019). Indeed, race and gender hate speech has become an increasingly important issue in social media, as it stands for 50% of the targets of hate speech in Twitter (Silva et al., 2016). Code and pre-trained models are available at `https://github.com/CPerelloC/UA-SemEval`.

2 Hate Speech Detection System

In this section we present our hate speech detection model. The main goal of our model is to identify hate speech given a piece of text, in this case a tweet. A high-level overview of our model is presented in Section 2.1 and the set of features that are used in our model are described in Section 2.2.

2.1 Model

Our model consists of a linear classifier based on Support Vector Machines (SVM), which have proved to provide competitive results in text categorization since their conception (Joachims, 1998). The SVM classifier is trained on tweets containing hate speech annotations. During training, the model is fed with features relevant to hate speech detection. Then, in the test phase the goal of our model is to classify unannotated tweets with the categories learned during the training phase. In the following section we describe the main features used in our SVM classifier.

2.2 Features

For the main task A (hate speech detection), we distinguish three groups of features[2]:

- *Bag-of-n-grams*: Bag-of-n-grams features, which have been already used for hate speech detection (Chatzakou et al., 2017), are often reported to be highly predictive and can be combined with other features to improve performance. We make use of unigrams, bigrams and trigrams, represented in the feature vectors by their frequency in a tweet.

- *Sentiment analysis*: Hate speech and sentiment analysis are closely related, and we can assume that negative sentiment usually pertains to a hate speech message (Schmidt and Wiegand, 2017). To integrate this feature into our model, we simply add the output of a pre-trained sentiment analysis classifier.

- *Word embeddings*: Word embeddings are low-dimensional vector representations of words and are used extensively in natural language processing (Goldberg, 2016). In particular, Bayot and Gonçalves (2016) showed that word embeddings provide a useful generalization signal in text classification when used in a similar setting. In our case, we add the average of the embeddings in a tweet as an additional feature in our SVM classifier.

For task B, we use two simple extra features with specific information about each subtask:

- For *target classification* (individual or group), we use the count of the plural nouns in the tweet as a feature.

- For *aggressive behaviour*, we use the count of the insults in the tweet as a feature. We hypothesize that a high level of insults may involve violent behaviour. To this end, we filter a database from insults collected at `https://hatebase.org/`.

2.3 Feature selection

One of the main issues in text classification is the high dimensionality of the feature space (Yang and Pedersen, 1997). For instance, over 300K and 150K features were initially obtained using the bag-of-n-grams features alone on, respectively, the English and Spanish training sets from Task A (see Section 3.1). Besides the computational cost to train a model with such a large amount of features, an additional issue is the noise that could be introduced by including many irrelevant features. Thus, it is generally desirable to reduce the feature space, without sacrificing classification accuracy.

The feature selection method used in our system is based on word frequency, understanding a word as an n-gram. The system first delimits the n-grams by a frequency number to significantly reduce the feature space before preparing the vectors for the SVM. Then, highly sparse features (i.e. containing zero in more than 99.9% of the samples) are removed.[3]

[2]We use an extra standard feature to these three groups, the *length of the tweet* in words.

[3]This was achieved by leveraging the *VarianceThreshold* tool from *scikit-learn* (Pedregosa et al., 2011): `https://scikit-learn.org/stable/modules/feature_selection.html`

3 Evaluation

In this section we describe the experimental setup (Section 3.1) of our system along with the results obtained (Section 3.2), including a brief analysis of errors detected in the evaluation phase (Section 3.3).

3.1 Experimental setup

In the following we present the datasets considered, details about the text preprocessing and feature selection procedures, the pre-trained models used as part of our model, and how parameter tuning was performed.

Datasets. We used the two datasets made available as part of the SemEval-2019 Task 5 competition: one for English and another for Spanish. The datasets consist of training, development and test splits. For English, the number of tweets for each split is 9100, 1000 and 2971 for training, development and test, respectively. Conversely, the Spanish splits contain 4600, 500 and 1600 tweets.

Preprocessing. Each tweet is tokenized using the *spaCy* NLP library[4]. We experimented with various preprocessing variants and decided to work with raw words as tokens (i.e., without applying lemmatization), removing punctuation and URLs but keeping emojis and stopwords (pronouns and articles can be relevant in the context of hate speech classification).

Feature selection. As explained in Section 2.3, a feature selection procedure is applied on the n-gram features to reduce their noise and size. After the feature selection is performed for the bag-of-n-grams features, the featured space is reduced from 336,669 to 4,605 in English task A, and from 177,003 to 4,217 in Spanish task A.

Pre-trained models. Regarding sentiment analysis, we used as features the polarity [-1.0, 1.0] and the subjectivity [0.0, 1.0] of a tweet according to *TextBlob*[5] (Loria et al., 2014). Note that Textblob is only optimized for English input and was not used for the Spanish tasks. We leave the exploration of Spanish sentiment analysis systems for future work.

As far as word embeddings are concerned, we made use of Spanish and English 100-dimensional FastText word embeddings (Bojanowski et al., 2017) trained on two large Twitter corpus from Spain and United States, respectively (Barbieri et al., 2016).

Parameter tuning. We experimented with several kernels and parameter configurations to train the Support Vector Machines, including polynomial and linear kernels. Since our system is trained with a large amount of features, it is hard to find an optimal parameter configuration for the polynomial kernel. Therefore, we decided to use a linear kernel, as the SVM training was faster and implied tuning less parameters. We fine-tuned the C parameter of the SVM using as validation the development set of the task. This parameter tuning was performed using bag-of-n-grams as features and on the Spanish dataset only. The value of C that achieved the highest accuracy in the development set was $C = 2^{-5}$ for Task A and Task B-target classification, and $C = 3$ for Task B-aggressive behaviour, which were fixed across all experiments.

3.2 Results

In the following we present our results for Task A (Section 3.2.1) and Task B (Section 3.2.2).

3.2.1 Task A

Task A consists of detecting hate speech (HS) against women or immigrants in the text. Systems were evaluated according to standard classification metrics such as accuracy and macro-F1 score.

Table 1 shows our Task A results in the development and evaluation sets comparing different sets of features described in Section 2.2. As can be observed in the table, the highest accuracy and macro-F1-score obtained in the development phase were, respectively, 78.4 and 77.9 using all features (i.e., n-grams, tweet length and word embeddings for Spanish) and 72.8 and 72.0 with n-grams for English (the same features including sentiment analysis in this case). The sentiment analysis feature provided a small improvement when combined with n-grams on the English development set, but had a negligible influence on the test set. In general, except for the word embeddings which seem to generalize better, all features performed close to a random baseline in English. A further analysis should be required to explain

[4] https://spacy.io/

[5] TextBlob is a public Python library for processing textual data that provides an API for common NLP tasks such as sentiment analysis: https://textblob.readthedocs.io/en/dev/index.html

Features	English				Spanish			
	Dev		Test		Dev		Test	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1
All	**72.8**	**72.0**	50.1	48.1	**78.4**	**77.9**	**73.1**	**72.2**
N-grams	72.1	71.5	50.0	48.0	77.2	76.6	73.0	71.9
N-grams and sent. analysis∗	72.5	71.8	50.1	48.0	77.4	76.8	73.0	71.9
Word embeddings	65.3	60.1	**57.5**	**56.9**	63.6	56.3	65.9	55.0
SVC baseline	-	-	*49.2*	*45.1*	-	-	*70.5*	*70.1*
MFC baseline	-	-	*57.9*	*36.7*	-	-	*58.8*	*37.0*

Table 1: Task A results using different sets of features. The row marked with ∗ was submitted to the task.

the difference between development and test results, which affected most participating systems. Some possible explanations are discussed in the Analysis section (Section 3).

Unlike in English, in Spanish our system obtains the best result with the configuration that performed best in the development set. Our official submission (n-grams and tweet length as features) ranked sixth in terms of macro-F1 and second in terms of accuracy among all 40 participating systems. In the English task, with the addition of word embeddings as feature, our system would have ranked third in terms of macro-F1.

3.2.2 Task B

Task B consists of identifying the target harassed as individual or generic (TR), and to classify hateful tweets as aggressive or not aggressive (AG). In addition to the individual macro-F1 scores for these two subtasks (i.e., TR and AG), two global scores based on the average macro-F1 scores and Exact Match Ratio (EMR) (Kazawa et al., 2005) are reported. The EMR score measures the percentage of instances which are correctly labeled in all subtasks, i.e., HS (hate speech), TR (target) and AG (aggressiveness). As previously explained, our official submission consisted of n-grams and sentiment analysis features, with the addition of the two extra features mentioned in Section 2.2: a count of plurals in each tweet for TR and a count of insults for AG.

Table 2 displays the results of our system on Task B. As can be observed, results for TR are better for Spanish than English, which could be attributed to the fact that Spanish uses more plural forms than English. Regarding AG, the reason could be that the insult database was more accurately filtered for Spanish than English. These results, however, show the general trend of participating systems in the task.

Finally, we noted that training only on the portion of tweets where hate speech is present is beneficial. In our official submission we used all tweets for training, irrespective of whether they were hateful or not. Using all tweets for training was clearly adding a lot of noise to the training, and without it, a significant increase in the performance was obtained. Table 3 shows the results using the full training set and the training set including tweets considered as hateful. As an example, in the Spanish test set, the macro-F1-scores using only hateful tweets for training were 92.8 and 87.8, which means an absolute improvement of 16.9 and 14.3 percentage points for target classification and aggressive behaviour, respectively.

3.3 Analysis

When analyzing the errors of our system, we found a number of cases where irony was present. It is worth noting that sometimes hate speech is expressed through irony, and therefore does not imply an aggressive behaviour. Moreover, offensive language does not necessarily imply hate speech, which poses an additional challenge to these systems. Here are some sample tweets of hate speech without aggressive behaviour from the development set:

> "Say it loud, say it clear, illegal #immigrants are not welcome here."

> "Poland: our country is safe because we haven't taken in refugees"

Finally, given the disparity of results between English development and test sets, we analyzed possible causes for this behaviour. In Task A, we obtained the best performance by only using word embeddings. One of the reasons for these results

	F1(HS)	F1(TR)	F1(AG)	F1(avg)	EMR
English Dev	71.8	72.7	60.9	68.5	56.9
English Test	48.0	68.2	54.4	56.8	31.2
Spanish Dev	77.9	80.6	81.6	80.0	68.4
Spanish Test	72.2	75.9	73.5	73.9	62.9

Table 2: Task B results in the development and evaluation phases.

Training	**English Test**		**Spanish Test**	
	F1(TR)	F1(AG)	F1(TR)	F1(AG)
Full	68.2	54.4	75.9	73.5
Only hateful	**88.0**	**70.9**	**92.8**	**87.8**

Table 3: Macro-F1 results in Task B by using different types of training data.

could be that, in the development set 64.8% of the vocabulary of the test set was present in the training set, whereas in the test set only 54.8% of the vocabulary overlapped with the vocabulary of the training set. This reduction in the overlapping vocabulary between training and test handicaps the performance of n-gram based systems, which heavily relies in vocabulary overlap. Word embeddings are less affected by this condition since they can capture synonymy relations and therefore are able to generalize better. This could explain why using word embeddings alone attained the best performance in this experiment, as the n-grams were not helpful.

4 Conclusion and future work

In this paper we described our system presented at SemEval 2019 Task 5. The system follows a traditional machine learning approach based on feature engineering, making use of n-grams, sentiment analysis and word embeddings as its main features. The results obtained show how word embeddings, when combined with n-grams and sentiment analysis, can improve the performance of the system. In Spanish task A, our proposed system obtained a remarkable macro-F1 score of 72.5 (sixth highest) and an accuracy of 73.6 (second highest). In view of these results, we have achieved our objective of building a strong baseline for hate speech detection.

Future directions of this work include incorporating users' features to the model, studying how the pronouns and the context of the tweet may affect hate speech classification, and comparing the resulting system with deep neural network approaches, which have recently gained popularity in text classification tasks.

Acknowledgments

We would like to thank Miguel Camacho and the Hate Crime National Office in Spain for their support.

References

Francesco Barbieri, German Kruszewski, Francesco Ronzano, and Horacio Saggion. 2016. How cosmopolitan are emojis?: Exploring emojis usage and meaning over different languages with distributional semantics. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 531–535. ACM.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Roy Bayot and Teresa Gonçalves. 2016. Author profiling using svms and word embedding averages. In *Proceedeings of the International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*. CEUR.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics*, 5(1):135–146.

Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Emiliano De Cristofaro, Gianluca Stringhini, and Athena Vakali. 2017. Mean birds: Detecting aggression and bullying on twitter. In *Proceedings of the 2017 ACM on web science conference*, pages 13–22. ACM.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. *arXiv preprint arXiv:1703.04009*.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th international conference on world wide web*, pages 29–30. ACM.

Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Hideto Kazawa, Tomonori Izumitani, Hirotoshi Taira, and Eisaku Maeda. 2005. Maximal margin labeling for multi-topic text categorization. In *Advances in neural information processing systems*, pages 649–656.

Steven Loria, P Keen, M Honnibal, R Yankovsky, D Karesh, E Dempsey, et al. 2014. Textblob: simplified text processing. *Secondary TextBlob: Simplified Text Processing*.

Vinita Nahar, Sayan Unankard, Xue Li, and Chaoyi Pang. 2012. Sentiment analysis for effective detection of cyber bullying. In *Asia-Pacific Web Conference*, pages 767–774. Springer.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153. International World Wide Web Conferences Steering Committee.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Leandro Silva, Mainack Mondal, Denzil Correa, Fabrício Benevenuto, and Ingmar Weber. 2016. Analyzing the targets of hate in online social media. In *Proceedings of the Tenth International AAAI Conference on Web and Social Media*.

Yiming Yang and Jan O Pedersen. 1997. A comparative study on feature selection in text categorization. In *Proceedings of the International Conference on Machine Learning*, volume 97, page 35.

UNBNLP at SemEval-2019 Task 5 and 6: Using Language Models to Detect Hate Speech and Offensive Language

Ali Hakimi Parizi, Milton King and **Paul Cook**
Faculty of Computer Science, University of New Brunswick
Fredericton, NB E3B 5A3, Canada
ahakimi@unb.ca, milton.king@unb.ca, paul.cook@unb.ca

Abstract

In this paper we apply a range of approaches to language modeling — including word-level n-gram and neural language models, and character-level neural language models — to the problem of detecting hate speech and offensive language. Our findings indicate that language models are able to capture knowledge of whether text is hateful or offensive. However, our findings also indicate that more-conventional approaches to text classification often perform similarly or better.

1 Introduction

SemEval 2019 Task 5 focuses on detecting hate speech in social media text, while Task 6 considers identifying offensive language. Despite the differences between hate speech and offensive language, each of these tasks can be viewed as a binary classification problem. In each case, gold-standard training data is provided on which supervised approaches can be trained.

In this paper we consider whether approaches to classification based on language models — including word- and character-level neural language models, as well as more-conventional (word-level) n-gram language models — are able to distinguish between hateful and not hateful, and offensive and not offensive, language. We find that these approaches outperform a most-frequent class baseline, indicating that language models can capture some knowledge of whether text is hateful or offensive. However, for task 5 — for which the testing data was made available for follow up experiments — we also find that more-conventional approaches to supervised classification, such as naive Bayes and fastText (Joulin et al., 2017), often give similar or better results.

2 Related Work

Social media such as Twitter and Facebook are widely used, and hate speech has become prevalent in these platforms. In response to this, a substantial amount of research has recently focused on detecting such disturbing comments.

Prior work on hate speech and offensive language detection has mostly focused on supervised machine learning techniques (Mathur et al., 2018; Davidson et al., 2017). A recent shared task on identifying aggression in social media (Kumar et al., 2018) observed there is no significant difference between the performance of neural networks and linear classifiers. Furthermore, this shared task received just one lexicon-based approach, and its performance was not promising. Moreover, all of these approaches required expensive feature engineering and pre-processing.

Instead of engineering specific features to use in supervised classifiers, we can instead employ language models to model the type of text we want to detect. Language models have previously been applied for the purpose of text classification (e.g., Bai et al., 2004; Howard and Ruder, 2018). Among different types of language models, recurrent neural network (RNN) language models with LSTM and GRU units have shown promising results for sequence modeling (Mikolov et al., 2012). To the best of our knowledge, RNN language models have not been widely used for detecting hate speech or offensive language. The most closely related work is that of Mehdad and Tetreault (2016), which used both word-level and character-level RNNs to detect abusive language. In their experiments, Mehdad and Tetreault found that character-level language models outperformed word-level language models. A further advantage of character-level language models is that they are able to model out-of-vocabulary words (Mikolov et al., 2012).

3 Task 5: HatEval

SemEval 2019 Task 5 includes two sub-tasks: (A) detecting hate speech in English and Spanish tweets, and (B) classifying hate speech tweets as aggressive or not, and as targeting an individual or group. We only consider sub-task A. In Section 3.1, we describe the approaches we considered for this task, and in Section 3.2 we present our results.

3.1 Approaches

3.1.1 Word-level LMs: n-gram and LSTM

For each language, we grouped the training instances based on their gold standard labels — giving us two corpora per language — with one consisting entirely of hateful tweets, and the other consisting of not hateful tweets. For each language, we then trained two language models (LMs), one on the hateful instances, and the other on the non-hateful instances. Given a test instance, we calculate the probability of the tweet under each LM. If the LM that was trained on the hateful text gave a higher probability then we labeled the instance as hateful, otherwise we labeled it as non-hateful. The two word-level LMs that we considered are an n-gram LM and a long short-term memory (LSTM) LM. The n-gram model was a 3-gram model with Kneser-Ney smoothing trained using Kenlm (Heafield et al., 2013) with its default settings. We tuned the parameters for the LSTM model on the English development dataset using grid search. Specifically we considered the following settings for the embedding size (256, 512, 1024), number of hidden units (128, 256, 512), and number of epochs (1,2,3,4,5). The final parameters for the LSTMs used on the test datasets were 1 hidden layer, an embedding size of 1024, 128 hidden units, and they were trained using a batch size of 2 and 1 epoch.

3.1.2 Character-level LM

This approach is the same as the previous word-level LM approach, except that character-level, as opposed to word-level, LMs are trained. We use a publicly available TensorFlow implementation of a character-level RNN language model.[1] The following parameters are used: a two-layer GRU with one-hot character embeddings and a hidden layer size of 64 dimensions. The batch size, learning rate, and dropout are set to 20, 0.002, and 0, respectively. The hidden layer size and unit were tuned on the development data. Specifically we considered hidden layer sizes of 64, 128, and 256, and an LSTM and GRU for the unit. The other parameters are their default settings.

3.1.3 Neural LMs with Class Token

In the previous approaches, two separate LMs are trained — one on the hateful tweets, the other on the non-hateful ones. In contrast, in this approach a single LM is trained.

We append a special token to the end of each tweet in the training data representing its gold standard class. We randomly shuffle the order of the tweets in the training data, and then train a LM model on them. At test time, we feed a tweet (which has not been augmented with a special token indicating its class) to the LM. We then query the LM for the probability of the special tokens representing the hateful and non-hateful classes, and classify the tweet as the class corresponding to the special token with higher probability.

We consider both word-level and character-level neural LMs for this approach. For the word-level LM we again use an LSTM. We performed a grid search to tune its parameters using the English development dataset. Specifically we considered the following settings for the number of layers (1,2), embedding size (128, 256, 512), number of hidden units (128, 256, 512), and number of training epochs (1,2,3). The final model consisted of two hidden layers, an embedding size of 512, 128 hidden units, and was trained using a batch size of two for three epochs. For the character-level LM we use the same model as in Section 3.1.2, with the same parameter settings.

3.1.4 Baselines

In addition to the most-frequent class and SVC baselines provided by the shared task (Basile et al., 2019), we also compare our approaches against multinomial naive Bayes[2] and fastText (Joulin et al., 2017). We use the default settings for fastText, and do not attempt to tune it to this task.

[1]https://github.com/crazydonkey200/tensorflow-char-rnn

[2]Note that the likelihood term in multinomial naive Bayes corresponds to a unigram LM for each class. As such it is similar to the LM-based approaches we consider, but also incorporates a class prior.

	English								Spanish							
	Dev				Test				Dev				Test			
Method	F	P	R	A	F	P	R	A	F	P	R	A	F	P	R	A
n-gram	**0.70**	0.71	0.71	**0.70**	0.45	**0.60**	**0.55**	0.49	0.71	0.72	0.72	0.71	0.66	0.67	0.68	0.66
LSTM	0.60	0.61	0.61	0.60	0.48	0.53	0.52	0.49	0.68	0.68	0.68	0.69	0.64	0.64	0.64	0.65
Char	**0.70**	0.71	0.71	**0.70**	0.45	0.57	0.54	0.49	0.67	0.68	0.67	0.68	0.66	0.66	0.66	0.67
LSTM+CT	0.50	0.54	0.54	0.51	**0.49**	0.53	0.53	0.50	0.55	0.56	0.56	0.56	0.52	0.57	0.56	0.53
Char+CT	0.52	0.53	0.53	0.53	0.47	0.47	0.47	0.49	0.53	0.54	0.53	0.55	0.48	0.48	0.48	0.51
NB	**0.70**	**0.73**	**0.72**	**0.70**	0.41	**0.60**	0.54	0.47	**0.74**	**0.77**	**0.73**	**0.75**	0.69	**0.70**	0.69	**0.71**
fastText	0.64	0.66	0.64	0.66	0.47	0.59	**0.55**	0.50	0.70	0.72	0.70	0.71	**0.70**	0.69	0.70	0.70
SVC	-	-	-	-	0.45	**0.60**	**0.55**	0.49	-	-	-	-	**0.70**	**0.70**	**0.71**	**0.71**
MFC	0.36	0.29	0.50	0.57	0.37	0.29	0.50	**0.58**	0.36	0.28	0.50	0.56	0.37	0.29	0.50	0.59

Table 1: Macro average F1-score (F), macro average precision (P), macro average recall (R), and accuracy (A) on Task 5 subtask A using word-level n-gram and LSTM LMs, a character-level LM (Char), a word-level LSTM and character-level LM augmented with a special class token (LSTM+CT and Char+CT), multinomial naive Bayes (NB), and fastText on the development and test sets. The SVC and most-frequent class (MFC) baselines provided by the shared task are also shown. The best result for each language, dataset, and evaluation measure is shown in boldface.

3.2 Results

The English and Spanish training sets contain 9,000 and 4,500 tweets, respectively, labelled as being hateful or non-hateful. The development and test sets contain 1,000 and 2,971 tweets, respectively, for English, and 500 and 1,600 tweets, respectively, for Spanish. Further details of the datasets are provided in Basile et al. (2019).

Results are shown in Table 1.[3] The character-level LM, which performed best on the development data, corresponds to our official submission for the shared task. In terms of F-score, for both languages, all LM-based approaches outperform the most-frequent class baseline. This indicates that LMs are able to capture information about whether a tweet is hate speech or not. However, more-conventional approaches to classification perform similarly to, or better than, the LM-based approaches. Focusing on the test data, for English, although the LSTM with class token approach achieves the best F-score of 0.49, fastText achieves only a slightly lower F-score of 0.47. For Spanish, fastText and SVC achieve the best F-score of 0.70, while the best LM-based approaches, the n-gram and character LMs, obtain an F-score of 0.66.

4 Task 6: OffensEval

SemEval 2019 Task 6 includes 3 sub-tasks. In contrast to Task 5, we participated in all subtasks of Task 6. Sub-task A is a binary classification task to determine if a tweet is offensive or non-offensive. Sub-task B is to classify tweets that are offensive into two groups of targeted — a post containing an insult or threat to an individual, a group, or others — or untargeted — a post containing non-targeted profanity and swearing. Finally, sub-task C is a three-way classification task in which targeted tweets (from sub-task B) are classified as targeting an individual, group of people, or other.

Because of the similarities between hate speech and offensive language, we apply approaches that we used for Task 5 to Task 6. We used the development data for Task 5 for model tuning and selection, and only considered the three best models for Task 6: the word-level n-gram and LSTM language models and the character-level language model. We describe these approaches in Section 4.1 and report results over the test data in Section 4.2.[4]

4.1 Approaches

4.1.1 Word-level LMs: n-gram and LSTM

Similar to Task 5, for each sub-task of Task 6, we group the training instances based on their gold-standard classes. We then train one LM on the documents from each class. I.e., in the case of sub-task A we train one LM on tweets labeled offensive, and another LM on tweets labelled non-offensive. At test time we measure the probability

[3]The SVC baseline was only provided for the test data.

[4]We do not report results for the development data because we used Task 5 data for model tuning and selection.

	Sub-task A		Sub-task B		Sub-task C	
Method	F	A	F	A	F	A
n-gram	**0.62**	0.66	0.45	0.50	**0.43**	0.44
LSTM	0.55	0.59	0.54	0.73	0.40	**0.48**
Char	0.59	0.63	**0.61**	0.88	-	-
MFC	0.42	**0.72**	0.47	**0.89**	0.21	0.47

Table 2: Macro-average F1-score (F) and accuracy (A) for each sub-task of Task 6 using word-level n-gram and LSTM LMs, a character-level LM (Char), and a most-frequent class baseline (MFC). The best result for each sub-task and evaluation measure is shown in boldface.

of a test tweet under each LM, and then classify it as the class corresponding to the LM giving the highest probability. Note that sub-task C is a three-way, as opposed to binary, classification task, and so we therefore train three LMs for this sub-task.

We consider the same word-level LMs as for Task 5 — an n-gram LM and an LSTM LM. We tuned the parameters for the LSTM on the English development dataset of Task 5, sub-task A, using grid search as described in Section 3.1.1. We did not further tune this model to Task 6. For the n-gram model we again use a 3-gram model as described in Section 3.1.1.

4.1.2 Character-level LM

We apply character-level LMs to each sub-task of Task 6 in the same manner as we use the word-level LMs described above. We use the same character-level LM as for Task 5, described in Section 3.1.2, with the same parameter settings. We do not attempt to further tune the parameters to Task 6.

4.2 Results

Details of the training and test data can be found in Zampieri et al. (2019). Results for the LM-based approaches described above, as well as a most-frequent class baseline, are presented in Table 2. In terms of F1-score, for each sub-task, each LM-based approach outperforms the most-frequent class baseline, with the exception of the n-gram LM on sub-task B.[5] These results, and in particular those on sub-task A, demonstrate that LMs can capture knowledge about whether text is offensive.

On sub-task A the n-gram LM achieved the best F1-score, while on sub-task B the character-level LM did. One reason for his could be differences in the size of the training data. For sub-task A, we use all of the training data (13,240 instances) to train our models. However, for sub-task B, we are limited to just those tweets that were labeled as offensive. There are only 4,400 such tweets. Although this reduction in the amount of training data caused the F1-score of the word-level LM-based approaches to decrease, the amount of training data seems to still be sufficient to train a character-level LM. The F1-scores on sub-task C are lower than for the other sub-tasks. One possible explanation for this relatively poor performance is that the training data for sub-task C is smaller than that for the other sub-tasks (3,876 instances), because the sub-task C training data is a subset of that for sub-task B.

5 Conclusions

In this paper we employed language models to the problems of detecting hate speech and offensive language in social media text. We considered a range of approaches to language modeling including word-level n-gram and neural language models, and a character-level neural language model. Our results indicated that language model-based approaches are able to capture knowledge of whether text is hateful or offensive. However, further experiments on identifying hate speech indicated that more-conventional approaches to text classification often perform comparably or better.

In this paper we only considered language models trained on the training data provided for the shared tasks. In future work, we intend to consider pre-training language models on other corpora (e.g., Twitter corpora) in an effort to improve the performance of language model-based approaches to detecting hate speech and offensive language.

[5]Due to an error in our submission for the character-level model for sub-task C, we did not receive results for this approach on this sub-task.

References

Jing Bai, Jian-Yun Nie, and François Paradis. 2004. Using language models for text classification. In *Proceedings of the Asia Information Retrieval Symposium (AIRS)*.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *InProceedings of ICWSM*, pages 512–515.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

Ritesh Kumar, Atul Kr Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 1–11.

Puneet Mathur, Rajiv Shah, Ramit Sawhney, and Debanjan Mahata. 2018. Detecting offensive tweets in hindi-english code-switched language. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, pages 18–26.

Yashar Mehdad and Joel Tetreault. 2016. Do characters abuse more than words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303.

Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. 2012. Subword language modeling with neural networks. *preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)*, 8.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. A Hierarchical Annotation of Offensive Posts in Social Media: The Offensive Language Identification Dataset. In *arxiv preprint*.

UTFPR at SemEval-2019 Task 5: Hate Speech Identification with Recurrent Neural Networks

Gustavo Henrique Paetzold[1], Shervin Malmasi[2], Marcos Zampieri[3]
[1]Universidade Tecnológica Federal do Paraná, Toledo-PR, Brazil
[2]Harvard Medical School, Boston, United States
[3]University of Wolverhampton, Wolverhampton, United Kingdom
ghpaetzold@utfpr.edu.br

Abstract

In this paper we revisit the problem of automatically identifying hate speech in posts from social media. We approach the task using a system based on minimalistic compositional Recurrent Neural Networks (RNN). We tested our approach on the SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter (HatEval) shared task dataset. The dataset made available by the HatEval organizers contained English and Spanish posts retrieved from Twitter annotated with respect to the presence of hateful content and its target. In this paper we present the results obtained by our system in comparison to the other entries in the shared task. Our system achieved competitive performance ranking 7th in sub-task A out of 62 systems in the English track.

1 Introduction

Abusive and offensive content such as aggression, cyberbulling, and hate speech have become pervasive in social media. The widespread of offensive content in social media is a reason of concern for governments worldwide and technology companies, which have been heavily investing in ways to cope with such content using human moderation of posts, triage of content, deletion of offensive posts, and banning abusive users.

One of the most common and effective strategies to tackle the problem of offensive language online is to train systems capable of recognizing such content. Several studies have been published in the last few years on identifying abusive language (Nobata et al., 2016), cyber aggression (Kumar et al., 2018), cyber bullying (Dadvar et al., 2013), and hate speech (Burnap and Williams, 2015; Davidson et al., 2017). As evidenced in two recent surveys (Schmidt and Wiegand, 2017; Fortuna and Nunes, 2018) and in a number of other studies (Malmasi and Zampieri, 2017; Gambäck and Sikdar, 2017; ElSherief et al., 2018; Zhang et al., 2018), the identification of hate speech is the most popular of what Waseem et al. (2017) refers to as "abusive language detection sub-tasks".

This paper deals with the hate speech identification in English and Spanish posts from social media. We present our submissions to the SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter (HatEval) shared task. We participated in sub-task A which is a binary classification task in which systems are trained to discriminate between posts containing hate speech and posts which do not contain any form of hate speech. Our approach, presented in detail in Section 4, combines compositional Recurrent Neural Networks (RNN) and transfer learning and achieved competitive performance in the shared task.

2 Related Work

As evidenced in the introduction of this paper, there have been a number of studies on automatic hate speech identification published in the last few years. One of the most influential recent papers on hate speech identification is the one by Davidson et al. (2017). In this paper, the authors presented the Hate Speech Detection dataset which contains posts retrieved from social media labeled with three categories: OK (posts not containing profanity or hate speech), Offensive (posts containing swear words and general profanity), and Hate (posts containing hate speech). It has been noted in Davidson et al. (2017), and in other works (Malmasi and Zampieri, 2018), that training models to discriminate between general profanity and hate speech is far from trivial due to, for example, the fact that a significant percentage of hate speech posts contain swear words. It has been argued that annotating texts with respect to the presence of hate speech has an intrinsic degree of subjectivity (Malmasi and Zampieri, 2018).

Along with the recent studies published, there have been a few related shared tasks organized on the topic. These include GermEval (Wiegand et al., 2018) for German, TRAC (Kumar et al., 2018) for English and Hindi, and OffensEval[1] (Zampieri et al., 2019b) for English. The latter is also organized within the scope of SemEval-2019. OffensEval considers offensive language in general whereas HatEval focuses on hate speech.

Waseem et al. (2017) proposes a typology of abusive language detection sub-tasks taking two factors into account: the target of the message and whether the content is explicit or implicit. Considering that hate speech is commonly understood as speech attacking a group based on ethnicity, religion, etc, and that cyber bulling, for example, is understood as an attack towards an individual, the target factor plays an important role in the identification and the definition of hate speech when compared to other forms of abusive content.

The two SemEval-2019 shared tasks, HatEval and OffensEval, both include a sub-task on target identification as discussed in Waseem et al. (2017). HatEval includes the target annotation in its sub-task B with two classes (individual or group) whereas OffensEval includes it in its sub-task C with three classes (individual, group or others). Another important similarity between these two tasks is that both include a more basic binary classification task in sub-task A. In HatEval, posts are labeled as as to whether they contain hate speech or not and in OffensEval, posts are labeled as being offensive or not. As OffensEval considers multiple types of offensive contents, the hierarchical annotation model used to annotate OLID (Zampieri et al., 2019a), the dataset used in OffensEval, includes an annotation level distinguishing between the type of offensive content that posts include with two classes: insults and threats, and general profanity. This type annotation is used in OffensEval's sub-task B.

3 Task Description

HatEval (Basile et al., 2019) provides participants with annotated datasets to create systems capable of properly identifying hate speech in tweets written in both English and Spanish.

The training, development, trial, and test sets provided for English are composed of 9,000, 1,000, 100 and 3,000 instances, respectively. The training, development, trial and test sets provided for Spanish are composed of 4,500, 500, 100 and 1,600 instances, respectively. Each instance is composed of a tweet and three binary labels: One that indicates whether or not hate speech is featured in the tweet, one indicating whether the hate speech targets a group or an individual, and another indicating whether or not the author of the tweet is aggressive. HatEval has 2 sub-tasks:

- **Sub-task A:** Judging whether or not a tweet is hateful.
- **Sub-task B:** Correctly predicting all three of the aforementioned labels.

In this paper, we focus on Task A exclusively, for both English and Spanish. We participated in the competition using the team name UTFPR.

4 The UTFPR Models

The UTFPR models are minimalistic Recurrent Neural Networks (RNNs) that learn compositional numerical representations of words based on the sequence of characters that compose them, then use them to learn a final representation for the sentence being analyzed. These models, of which the architecture is illustrated in Figure 1, are somewhat similar to those of Ling et al. (2015) and Paetzold (2018), who use RNNs to create compositional neural models for different tasks.

As illustrated, the UTFPR models take as input a sentence, split it into words, then split the words into a sequence of characters in order to pass them through a character embedding layer. The character embeddings are passed onto a set of bidirectional RNN layers that produces word representations, then a second set of layers produces a final representation of the sentence. Finally, this representation is passed through a softmax dense layer that produces a final classification label.

For each language, we created two variants of UTFPR: one trained exclusively over the training data provided by the organizers (UTFPR/O), and another that uses a pre-trained set of character-to-word RNN layers extracted from the models introduced by Paetzold (2018) (UTFPR/W). The pre-trained model was trained for the English multi-class classification Emotion Analysis shared task

[1] `https://competitions.codalab.org/competitions/20011`

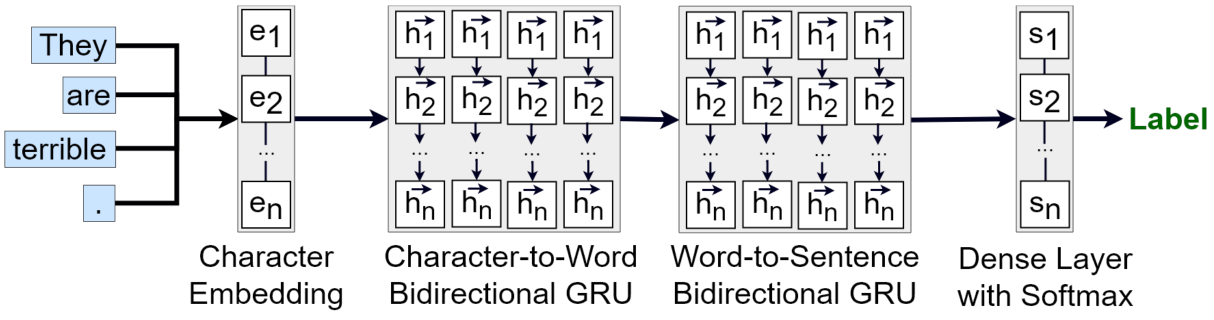


Figure 1: Architecture of the UTFPR models.

of WASSA 2018, which featured a training set of 153,383 instances composed of a tweet and an emotion label. This pre-trained model for English was used for the UTFPR/W variant of both languages, since we wanted to test the hypothesis that pre-training a character-to-word RNN on a large dataset for English can improve the performance of compositional models for both English and Spanish.

We use 25 dimensions for the size of our character embeddings, and two layers of Gated Recurrent Units for our bidirectional RNNs with 60 hidden nodes each and 50% dropout. We saved a model after each training iteration and picked the one with the lowest error on the development set. The UTFPR/W model went through the same training process as UTFPR/O, with the pre-trained character-to-word RNN layers being fine-tuned for the task at hand.

Table 1 showcases the F-scores obtained by the UTFPR systems on the trial set of Task A. Because of its superior performance, we chose to submit the UTFPR/W variants as our official entry.

	F-scores	
System	**English**	**Spanish**
UTFPR/O	0.509	0.601
UTFPR/W	0.570	0.665

Table 1: F-scores obtained for the trial set at HatEval Task A for both languages.

5 Results and Discussion

5.1 Shared Task Performance

Tables 2 and 3 feature the F-scores obtained by the UTFPR systems and the 3 best and worst performing systems at HatEval Task A for English and Spanish, respectively. Ultimately, the UTFPR/W systems submitted ranked 7th out of 62 valid submissions for English, and 31st out of 35 valid submissions for Spanish.

System	F-scores
FERMI	0.650
Panaetius	0.570
YNU_DYX	0.550
UTFPR/O	0.524
UTFPR/W	0.513
MELODI	0.350
INGEOTEC	0.350
INAOE-CIMAT	0.340

Table 2: F-scores obtained at HatEval Task A for the English language. At the top and bottom of the table are featured the top and bottom 3 systems submitted to the shared task, respectively.

System	F-scores
mineriaUNAM	0.730
Atalaya	0.730
MITRE	0.730
UTFPR/O	0.664
UTFPR/W	0.636
jhouston	.630
LU team	0.620
TuEval	0.620

Table 3: F-scores obtained at HatEval Task A for the Spanish language. At the top and bottom of the table are featured the top and bottom 3 systems submitted to the shared task, respectively.

One of the aspects we wanted to test with our participation in this shared task was the extent to which pre-training a character-to-word RNN over a larger dataset for an analogous task helped the models. Our results show that, even though using a pre-trained RNN considerably improved the performance of our models in the trial experiments, it actually compromised their performance for the

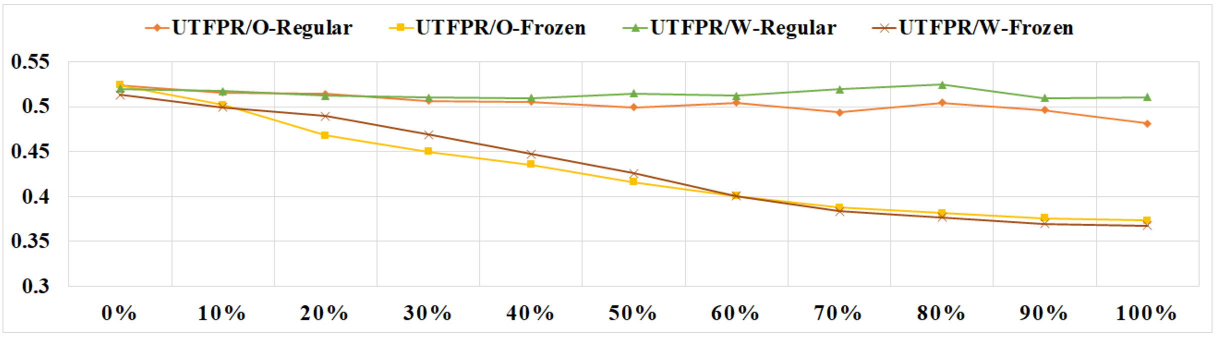


Figure 2: F-scores of our robustness experiments for English. The horizontal axis represents the proportion of noisy words in the input sentence, and the vertical axis the F-scores.

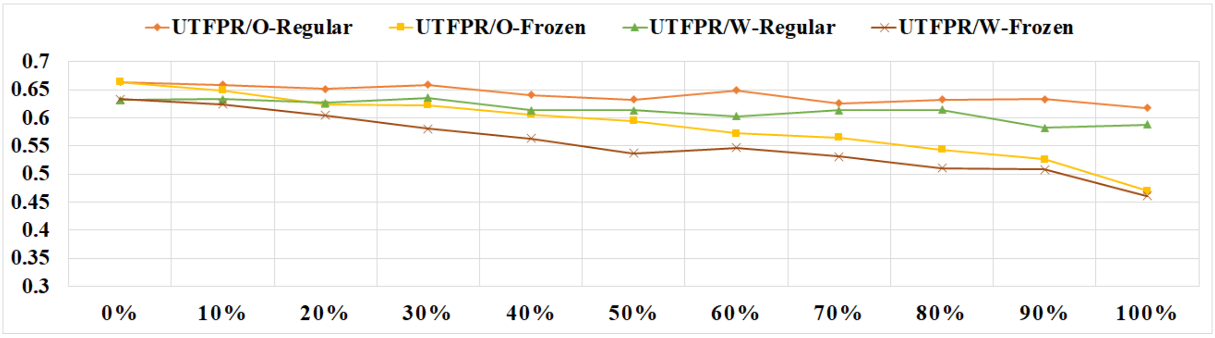


Figure 3: F-scores of our robustness experiments for Spanish. The horizontal axis represents the proportion of noisy words in the input sentence, and the vertical axis the F-scores.

test set a little. We believe that this was caused because the development set was more representative of the trial than the test set. Overall, submitting UTFPR/W instead of UTFPR/O cost us 2 ranks for English and 3 for Spanish.

5.2 Robustness Assessment

In order to test the robustness of the UTFPR systems, we had to generate different noisy versions of the test set with increasing volumes of noise artificially added to them.

To do so, we introduced a modification to $N\%$ of randomly selected words in each sentence in the datasets. The modifications could be either the deletion of a randomly selected character (50% chance) or its duplication (50% chance). We used $0 \leq N \leq 100$ in intervals of 10, resulting in a total of 11 increasingly noisy versions. The next step was to create "frozen" versions of the UTFPR models that act as if any word out of the training set's vocabulary is unknown. If a word of the test set is not present in the vocabulary of the training set, it produces a numerical vector full of 1's that represents an out-of-vocabulary word.

Figures 2 and 3 show the results obtained for English and Spanish, respectively. As it can be noticed, our compositional models are much more robust than the frozen alternatives, suffering very faint losses in F-score even when 100% of the words in the input sentence are noisy.

6 Conclusions

In this contribution, we presented the UTFPR systems submitted to the HatEval 2019 shared task. The systems are based on compositional RNN models trained exclusively over the training data provided by the organizers. We introduced two variants of our models: one trained entirely on the shared task's data (UTFPR/O), and another with a set of pre-trained character-to-word RNN layers fine-tuned to the task at hand (UTFPR/W). Our results show that, despite its simplicity, the UTFPR/O model attained competitive results for English, placing it 7th out of 62 submissions. Furthermore, the results of this shared task indicate that our models are very robust, being able to handle even substantially noisy inputs. In the future, we intend to test more reliable ways of re-using pre-trained compositional models.

Acknowledgements

We would like to thank the organizers of the HatEval shared task for providing participants with this dataset and for organizing this interesting shared task. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan V GPU used for this research.

References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Pete Burnap and Matthew L Williams. 2015. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet*, 7(2):223–242.

Maral Dadvar, Dolf Trieschnigg, Roeland Ordelman, and Franciska de Jong. 2013. Improving cyberbullying detection with user context. In *Advances in Information Retrieval*, pages 693–696. Springer.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 EMNLP*, pages 1520–1530. Association for Computational Linguistics.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 467–472.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive Language Detection in Online User Content. In *Proceedings of the 25th International Conference on World Wide Web*, pages 145–153. International World Wide Web Conferences Steering Committee.

Gustavo Paetzold. 2018. Utfpr at iest 2018: Exploring character-to-word composition for emotion analysis. In *Proceedings of the 9th EMNLP*, pages 176–181. Association for Computational Linguistics.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

Vista.ue at SemEval-2019 Task 5: Single Multilingual Hate Speech Detection Model

Kashyap Raiyani, Teresa Gonçalves, Paulo Quaresma and Vitor Beires Nogueira
Computer Science Department, University of Évora, Portugal
`(kshyp,tcg,pq,vbn)@uevora.pt`

Abstract

This paper shares insight from participating in SemEval-2019 Task 5. The main propose of this system-description paper is to facilitate the reader with replicability and to provide insightful analysis of the developed system. Here in Vista.ue, we proposed a single multilingual hate speech detection model. This model was ranked 46/70 for English Task A and 31/43 for English Task B. Vista.ue was able to rank 38/41 for Spanish Task A and 22/25 for Spanish Task B.

1 Introduction

According to the article (Bosco et al., 2017), nearly a quarter of a billion people, throughout the world, currently live in a country other than their place of birth. This is an increase of 41% from 2000 to 2015. This figure includes more than 21 million refugees often vulnerable and dissatisfied. Since 2015, Europe is facing an unprecedented refugee crisis, the by-effect of the Syrian civil war and the terrible living conditions in equatorial Africa. 1,300,000 people have generated this increased migration flow to Europe which can only but increase, putting European stable societies, so far, under pressure.

Therefore, the implications for the European society and the way we behave towards immigration, immigrant integration and social inclusion for newcomers and their children, are becoming more decisive and must be addressed either at a local or global level, considering a political and social perspective. While this phenomenon stimulates the generation and diffusion of hate speech and hate crimes, at the same time several initiatives are promoted, but they should be further improved to increase the awareness and empathy of receiving populations while avoiding polarization against immigrants.

Hate speech analysis and hate maps allow both a greater understanding of social phenomena linked to the integration of migrants, that more targeted actions to improve it. The integration of migrants is strongly linked to the new cultural context where they try to rebuild their lives. The process of acculturation depends on personal and social variables of the migrant, in large part in turn dependent on the cultural context of his/her origin, on the characteristics of the context of resettlement and on events occurring during this life period. The different migrants strategies firstly affect the different outcomes achieved. In particular, he can decide whether or not to maintain the cultural identity of origin and whether or not to establish and maintain new relationships within the new contest. This gives rise to four possible different outcomes: integration, assimilation, separation/segregation, marginalization (Berry, 1997).

1.1 Motivation

Data released by European Community about population change (Union, 2015) show that from the 1990s onwards natural population change had a diminishing role in EU demographic developments, while the role of net migration became increasingly important. In the period 2011 to 2013, net migration contributed more than 80% to total population growth, drawing an overall pattern of growth of EUs populations driven increasingly by changes in migratory flows, which hides a range of demographic situations among the EU Member States. Between 2004 and 2013, indeed the population of 11 EU Member States decreased, with the biggest reductions recorded in Germany and Romania, but a high overall increase in population numbers was recorded in the other countries like UK (a gain of 4.51 million inhabitants), Spain (3.96 million), France (3.54 million) and Italy (3.29 million). Among these countries, characterized by a negative natural population change, also compounded by negative net migration, Italy is affected by a negative natural change, that was completely offset by net migration which accounted for 108% of the total population change.

As a part of the motivation, we participated in the shared task named ”SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter”. Section 2 outlines the existing approaches in a systematic manner and the description of the task mentioned in Section 3. Paper also provides a short, comprehensive and structured overview of automatic hate speech detection in Section 4 followed my result comparison and conclusion in Section 5 and 6 respectively.

2 Related Work

For any text classification task, the most obvious information to utilize is surface-level features, such as a bag of words. Indeed, unigrams and larger n-grams are included in the feature sets by a majority of authors (Chen et al., 2012; Sood et al., 2012; Xu et al., 2012; Warner and Hirschberg, 2012; Van Hee et al., 2015). These features are often reported to be highly predictive. Still, in many works, n-gram features are combined with a large selection of other features. For example, in their recent work, (Nobata et al., 2016) report that while token and character n-gram features are the most predictive single features in their experiments, combining them with all additional features further improves performance.

Character level n-gram features might provide a way to attenuate the spelling variation problem often faced when working with user-generated comment text. For instance, the phrase ”ki11 yrslef a$$hole”, which is regarded as an example of hate speech, will most likely pose problems to token based approaches since the unusual spelling variations will result in very rare or even unknown tokens in the training data. While using Character level approaches, on the other hand, are more likely to capture the similarity to the canonical spelling of these tokens. Author (Mehdad and Tetreault, 2016) systematically compare character n-gram features with token n-grams for hate speech detection and found that character n-grams prove to be more predictive than token n-grams.

Apart from word and character based features, hate speech detection can also benefit from other surface features (Chen et al., 2012; Nobata et al., 2016), such as information on the frequency of URL mentions and punctuation, comment and token lengths, capitalization, words that cannot be found in English dictionaries, and the number of non-alpha numeric characters present in tokens.

Hate speech and sentiment analysis are closely related, and it is safe to assume that usually, negative sentiment pertains to a hate speech message. Because of this, several approaches acknowledge the relatedness of hate speech and sentiment analysis by incorporating the latter as an auxiliary classification. Author (Dinakar et al., 2012; Sood et al., 2012; Njagi et al., 2015) followed a multistep approach in which a classifier dedicated to detect negative polarity is applied prior to the classifier specifically checking for evidence of hate speech. Further, (Njagi et al., 2015) run an additional classifier that weeds out non-subjective sentences prior to the aforementioned polarity classification.

3 Task Description and Dataset

The main task (Basile et al., 2019) was to detect Hate Speech in Twitter toward two different targets, immigrants and women. The data were available in a multilingual perspective, English, and Spanish.

3.1 Task Description

The task was partition into two groups: Task A and Task B. Making a total of four subtasks (English/Spanish task A/B).

TASK A - Hate Speech(HS) Detection against Immigrants and Women: a two-class (or binary) classification where systems have to predict whether a tweet in English or in Spanish with a given target (women or immigrants) is hateful or not hateful.

TASK B - Aggressive behavior(AG) and Target Classification(TR): where systems are asked first to classify hateful tweets for English and Spanish (e.g., tweets, where Hate Speech against women or immigrants has been identified,) as aggressive or not aggressive, and second to identify the target harassed as individual or generic (i.e. single human or group).

A binary value (1/0) indicating if HS is occurring against one of the given targets (women or immigrants). If HS occurs (i.e. the value for the feature at point HS is 1), a binary value indicating if the target is a generic group of people (0) or a specific individual (1) denoted as TR. And if HS occurs (i.e. the value for the feature at point HS is 1), a binary value indicating if the tweeter is aggressive (1) or not (0) denoted as AG. Thus, making 3 columns (named HS, TR and, AG) for each tweet.

3.2 Dataset

As per detail provided by the organizing committee, all data for the competition were collected from Twitter and manually annotated mainly via the "Figur8 crowdsourcing platform". The Table 1 describes the distribution of the dataset.

Language	Task	Train	Dev	Test
English	A	9000	1000	3000
English	B	9000	1000	3000
Spanish	A	5000	500	1600
Spanish	B	5000	500	1600

Table 1: Task Dataset Distribution.

The Table 2 and 3 describes the Hate Speech Tweet data distribution/property over training and development dataset.

Task	Non-HS	HS
EN-A	5790	4210
EN-B	5790	1463 (TR=AG=0)
ES-A	2921	2579
ES-B	2921	315 (TR=AG=0)

Table 2: Task A/B Data Property of Non-HS/HS.

Lan	TR(AG=0)	AG(TR=0)	AG=TR=1
EN	984	1187	576
ES	86	498	1180

Table 3: Task B Data Property (HS=1).

4 System Description

This section will talk about the preprocessing of the data, the experimental setup, and the multilingual system architecture.

4.1 Tweet Preprocessing

Here, for EN/ES tweets, we are only removing "url" from each tweet. This is done with the help of regular expression.

```
r"http\S+", "url", tweet
```

4.2 Experimental Setup

Here, a common architecture is used for all the four subtasks. The only difference is the hyper-parameter. The Table 4 shows the experimental parameter values.

Task	Paramter	Value
EN/ES - A/B	batch_size	1
EN/ES - A/B	epochs	2
EN/ES - A/B	optimizer	Adam
EN/ES - A/B	validation_split	0.20

Table 4: Experimental Parameter.

4.3 Single Multilingual System Architecture

Author (Raiyani et al., 2018) have used simple feedforward dense architecture and able to achieve beyond the average result for finding aggression over social media (Facebook and Twitter). In particular, their model was able to stand the best performing model for English Tweets. Using a similar architectural concept, here, we are using a character-based dictionary. First of all, all the unique characters from the dataset are stored in the form of a dictionary. Then, using this dictionary, each character in the dataset are replaced by its key value. Thus, this transforms the dataset into an integer from the text. Finally, this integer data is further transformed into a binary array and fed to the Dense architecture. The Figure 1 shows the flowchart of system process. Where as the Figure 2 shows the Dense Architecture.

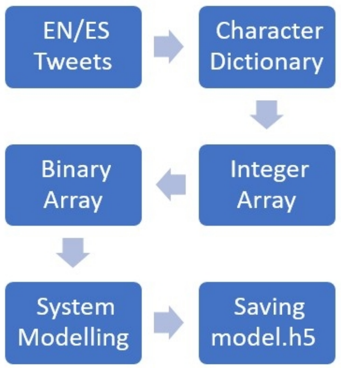


Figure 1: System Flowchart.

To store the intermediate character into the dictionary, pika library[1] was used. The number of unique characters found for English and Spanish

[1] https://pika.readthedocs.io/en/stable/

is respectively 169 and 172 (this also includes all the special character and emojis).

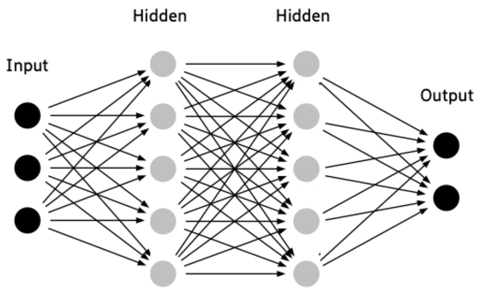


Figure 2: Feedforward Dense Architecture.

The architectural hyper parameter were selected based on trail and run. The same can be found in the Table 5 . The code of the entire task could be found in the online GitHub repository (Raiyani, 2019).

Task	Dense	Value	Activation
EN-A/B	layer 1	100	Relu
EN-A	layer 2	200	Sigmoid
EN-B	layer 2	200	Relu
ES-A/B	layer 1/2	50	Relu
EN/ES-A/B	layer 3	2	Softmax

Table 5: Architecture Parameter.

In the next section we will talk about the system performance and its global standing in the task.

5 Result Comparison and Discussion

The Table 6 shows the English task A average precision, recall, and F1 measure in reference to the baseline (SVC and MFC). The same for Spanish task A is found in the Table 7. The Table 8 shows the F1 measure over all the three parameter (namely, Hate Speech (HS), Target Classification(TR), and Aggressive(AG)). The ranking of task B is done using the value of Exact Match Ratio(EMR) (the evaluation formula could be found here[2]). The Table 9 shows the EMR value in reference to the baseline SVC and MFC.

The provided final ranking among all the subtasks are shown in Table 10.

6 Conclusion and Future Work

In this system description paper, we presented a single multilingual model for hate speech detection among immigrant and women. Through the

[2]https://competitions.codalab.org/competitions/19935

System	P	R	F1
Heigh	0.690	0.679	0.651
SVC	0.595	0.549	0.451
Vista.ue	0.483	0.488	0.420
MFC	0.289	0.500	0.367

Table 6: English - Task A Result.

System	P	R	F1
Heigh	0.734	0.741	0.730
SVC	0.701	0.707	0.701
Vista.ue	0.596	0.593	0.594
MFC	0.294	0.500	0.370

Table 7: Spanish - Task B Result.

System F1	Low	High	Obtain
EN B - HS	0.348	0.602	0.463
EN B - TR	0.372	0.752	0.596
EN B - AG	0.214	0.621	0.530
ES B - HS	0.370	0.761	0.573
ES B - TR	0.424	0.824	0.640
ES B - AG	0.413	0.760	0.578

Table 8: English/Spanish Task B F1 Result.

EMR	MFC	SVC	High	Obtain
EN B	0.580	0.308	0.580	0.284
ES B	0.588	0.605	0.635	0.536

Table 9: English/Spanish Task B EMR Result.

Task	System	Rank
EN A	SVC	35
	Vista.ue	46
	MFC	68
EN B	MFC	1
	SVC	27
	Vista.ue	31
ES A	SVC	21
	Vista.ue	38
	MFC	41
ES B	SVC	13
	Vista.ue	23
	MFC	18

Table 10: System Ranking.

system ranking, we can see that for task A of both the languages, the system is performing better than MFC baseline where on task B results could be improved. Further, We consider that our system can be grown, mainly due to the following facts: (1) The system does not count any NLP feature into account (2) Due to this, many hate tweets are missed. (3) Especially, for task B, features like Part of Speech (POS) tagging and Entity Extraction (EE) can improve the result. Lastly, how to address these aspects and generate a more accurate, comprehensive and fine-grained hate speech detection remains our further work.

Acknowledgments

The authors would like to thank COMPETE 2020, PORTUGAL 2020 Programs, the European Union, and ALENTEJO 2020 for supporting this research as part of Agatha Project SI & IDT number 18022 (Intelligent analysis system of open of sources information for surveillance/crime control).

References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

John W. Berry. 1997. Immigration, acculturation, and adaptation. *Applied Psychology*, 46(1):5–34.

Cristina Bosco, Viviana Patti, Marcello Bogetti, Michelangelo Conoscenti, Giancarlo Francesco Ruffo, Rossano Schifanella, and Marco Stranisci. 2017. Tools and resources for detecting hate and prejudice against immigrants in social media. In *SYMPOSIUM III. SOCIAL INTERACTIONS IN COMPLEX INTELLIGENT SYSTEMS (SICIS) at AISB 2017*, pages 79–84. AISB.

Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. 2012. Detecting offensive language in social media to protect adolescent online safety. In *Proceedings of the 2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust*, SOCIALCOM-PASSAT '12, pages 71–80, Washington, DC, USA. IEEE Computer Society.

Karthik Dinakar, Birago Jones, Catherine Havasi, Henry Lieberman, and Rosalind Picard. 2012. Common sense reasoning for detection, prevention, and mitigation of cyberbullying. *ACM Trans. Interact. Intell. Syst.*, 2(3):18:1–18:30.

Yashar Mehdad and Joel Tetreault. 2016. Do characters abuse more than words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303. Association for Computational Linguistics.

Dennis Njagi, Z Zuping, Damien Hanyurwimfura, and Jun Long. 2015. A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering*, 10:215–230.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 145–153, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Kashyap Raiyani. 2019. Single multilingual hate speech detection model. `https://github.com/kraiyani/Single-Multilingual-Hate-Speech-Detection-Model`.

Kashyap Raiyani, Teresa Gonçalves, Paulo Quaresma, and Vitor Beires Nogueira. 2018. Fully connected neural network with advance preprocessor to identify aggression over facebook and twitter. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 28–41. Association for Computational Linguistics.

Sara Owsley Sood, Elizabeth F. Churchill, and Judd Antin. 2012. Automatic identification of personal insults on social news sites. *J. Am. Soc. Inf. Sci. Technol.*, 63(2):270–285.

European Union. 2015. *People in the EU: who are we and how do we live?*, 2015 edition edition. Luxembourg.

Cynthia Van Hee, Els Lefever, Ben Verhoeven, Julie Mennes, Bart Desmet, Guy De Pauw, Walter Daelemans, and Véronique Hoste. 2015. Detection and fine-grained classification of cyberbullying events. In *Proceedings of the 10th Recent Advances in Natural Language Processing (RANLP 2015)*, Hissar, Bulgaria.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, LSM '12, pages 19–26, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from bullying traces in social media. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 656–666, Stroudsburg, PA, USA. Association for Computational Linguistics.

YNU NLP at SemEval-2019 Task 5: Attention and Capsule Ensemble for Identifying Hate Speech

Bin Wang, Haiyan Ding[*]
School of Information Science and Engineering
Yunnan University, Yunnan, P.R. China
hyding@ynu.edu.cn

Abstract

This paper describes the system submitted to SemEval 2019 Task 5: Multilingual detection of hate speech against immigrants and women in Twitter (hatEval). Its main purpose is to conduct hate speech detection on Twitter, which mainly includes two specific different targets, immigrants and women. We participate in both subtask A and subtask B for English. In order to address this task, we develope an ensemble of an attention-LSTM model based on HAN and a BiGRU-capsule model. Both models use fastText pre-trained embeddings, and we use this model in both subtasks. In comparison to other participating teams, our system is ranked 16th in the Subtask A for English, and 12th in the Subtask B for English.

1 Introduction

In recent years, the popularity of social networking and microblogging sites has increased, attracting more and more users. With this huge user base, social media will continue to release a large number of user-generated content. As the use of social media has grown, other undesirable phenomena and behaviors have emerged. Social media users often abuse this freedom to spread abuse or hateful posts or comments. In many cases, these user-generated content is inherently offensive or proactive, and users may have to deal with threats such as cyber attacks or cyberbullying, as well as other undesirable phenomena (Whittaker and Kowalski, 2015). So the problem of detecting and possibly limiting the spread of hate speech is becoming more and more important.

In order to solve the problem of abuse of language in social media platforms, some related research has been published, such as cyberbullying (Dadvar et al., 2013), hate speech (Warner and Hirschberg, 2012) and abusive language (Chen et al., 2012), most methods are based on surveillance methods (Schmidt and Wiegand, 2017). There are also some (racial discrimination) bias towards specific goals. In (Waseem and Hovy, 2016), the authors proposed a series of criteria based on critical race theory to identify racism and gender discrimination, they use n-gram models for research; Tulkens et al. studied racism detection in Dutch social media (Tulkens et al., 2016). A recent discussion of the challenge of identifying hate speech was proposed by Kumar et al. (Kumar et al., 2018). The results show that it is difficult to distinguish between open and covert attacks in social media.

SemEval 2019 Task 5 is proposed to identify hate speech about immigrants and women in Twitter for English or Spanish, and classify hate speech and judge whether the target is an individual or a group (Basile et al., 2019). Hate speech is often defined as any communication that attacks an individual or group through certain characteristics (such as gender, nationality, religion, or other characteristics) in social media platforms. This task gives us some text data from Twitter, we need to classify the content through computational analysis. The task has two subtasks, in which Subtask A is Hate speech detection for immigrants and women: It's a binary classification task, the system must judge whether a tweet with a specific goal (female or immigrant) in English or Spanish is hate speech; Subtask B is Aggressive behavior and target classification: This subtask is to classify the identified hate speech based on Subtask A, to judge whether it is aggressive or non-aggressive, and then to identify the target being harassed as an individual or group.

In this paper, we developped a system stacked two different neural network models: an attention-based model with LSTMs and an Capsule-based model with BiGRUs. We make some changes to

[*]Corresponding author

Hierarchical Attention Network to make it more suitable for this task, the detailed description of the Attention-LSTM model is provided in Section 2.2. Next, we build a BiGRU-Capsule model using the latest "Capsule" model proposed by (Sabour et al., 2017), the detailed description is provided in Section 2.3. In Section 2.4, we describe the use of stacking as ensemble. In Section 3.1, some details about data preprocessing for this task are described. In Section 3.2 and Section 3.3, the hyper-parameter setting and result analysis used in the whole experiment are introduced in detail.

2 Data and System Description

2.1 Data description

In this task, we only use the official training data set for training and trial data set to verify. In Subtask A, the purpose is to distinguish whether the tweet is hate speech, the data is divided into two categories(HS): 0 means non-hate speech, and conversely, 1 is hate speech. Similarly, in Subtask B, 0 is indicative of aggressiveness in categorizing hate speech(TR), 1 is non-aggressive, and in the goal of judging hate speech(AG), 0 means individual and 1 means group. In this task, we only participate in Subtask A and Subtask B in English. There are 9000 tweets in training data set, 1000 tweets in development data set, and 2971 tweets in final test data set. In the training data set, there are 5,217 labels are 0s and 3,783 labels are 1s in the label HS; in the label TR, 7659 labels are 0s and 1341 labels are 1s; and in the label AG 7440 labels are marked as 0s, 1560 are marked as 1s. Although the data of the label TR and the label AG are very unbalanced, since the ratio of 0 and 1 in the label HS is close to balance, we have not dealt with the data imbalance in this task.

2.2 Attention-LSTM Model

Here we have made some changes to HAN (Yang et al., 2017). The overall structure is shown in Figure 1. The replacement of BiGRU with LSTM (Hochreiter and Schmidhuber, 1997) is found to be significantly better than the original model for this task. The architecture of Attention-LSTM model is shown in Figure 1.

We use an LSTM to encode the sentences and to get annotations of words by summarizing information for word.

Not all words contribute equally to the expression of the emotion in the sentence. Emotion

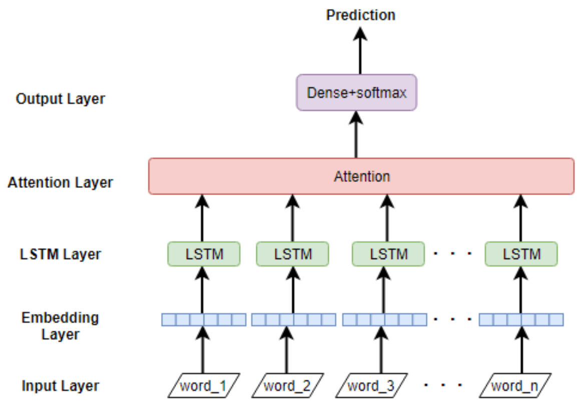


Figure 1: The architecture of Attention-LSTM Model.

greatly influences whether the sentence is hate speech, and also is helpful in identifying hate categories. There may be only a few words in a sentence that are crucial for the judgment of the goal of hate speech. So here we introduce the attention mechanism so that the system can better focus on words that are useful to identify hate speech, then it extracts those words and aggregates the representation of those important words to form a sentence vector.

First, we feed the word annotation h_i, and through a one-layer MLP to get a deeper representation u_i.

$$u_i = \tanh(W_s * h_i + b_s) \tag{1}$$

Then, we compute the similarity between u_i and word-level context vector u_s, and obtain a normalized weight α_i of importance by softmax function.

$$\alpha_i = \frac{exp(u_i^T * u_s)}{\sum_i exp(u_i^T * u_s)} \tag{2}$$

Finally, we compute the sentence vector s by a weighted sum of the word annotations h_i based on the normalized importance weights. s summarizes all the information of words in a context.

$$s = \sum_i \alpha_i * h_i \tag{3}$$

2.3 BiGRU-Capsule model

In order to improve the performance, in this system we use BiGRU and the latest capsule model (Sabour et al., 2017). The architecture of BiGRU-Capsule model is shown in Figure 2.

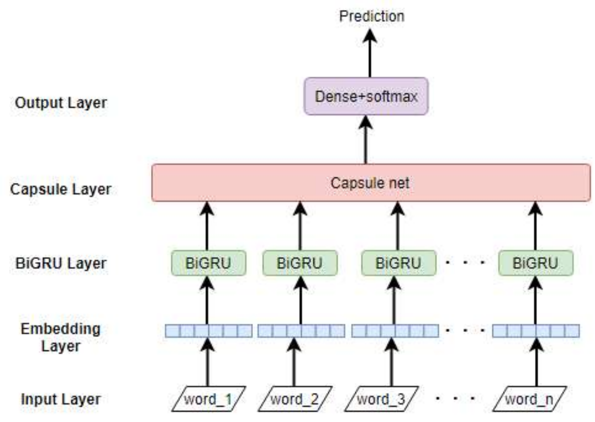


Figure 2: The architecture of BiGRU-Capsule Model.

First, we use the BiGRU layer to encode the sentences. As a variant of LSTM, GRU combines the Forget Gate and the Input Gate into a single Update Gate.

The bidirectional GRU is composed of two GRUs stacked one on top of the other. The output is determined by the state of the two GRUs.

In the capsule layer, the feature output by the previous BiGRU layer as an input to feed to the capsule network, to obtain deeper feature information. Capsule network was proposed by (Sabour et al., 2017), the main idea is to use neuron vectors instead of single neuron nodes of traditional neural networks, and finally train this new neural network by means of Dynamic Routing.

First, the "prediction vectors" $\widehat{u}_{j|i}$ are obtained by multiplying the output u_i of each capsule by a weight matrix W_{ij}.

$$\widehat{u}_{j|i} = W_{ij} * u_i \tag{4}$$

Then, all the "prediction vectors" are weighted summed to obtain the capsule s_j

$$s_j = \sum_i c_{ij} * \widehat{u}_{j|i} \tag{5}$$

where c_{ij} is the coupling coefficient between the capsules determined by "routing softmax", and the sum of the coupling coefficient of all the capsule is 1 in the layer.

Finally, we use the nonlinear "squashing" function to compress the length of the output vector of capsule between 0 and 1.

$$v_i = \frac{||s_j||^2}{1 + ||s_j||^2} \frac{s_j}{||s_j||} \tag{6}$$

where v_j is the output vector of capsule j.

2.4 Ensemble

Ensembling of several models is a widely used method to improve the performance of the overall system by combining predictions of several classifiers (Hansen and Salamon, 2002). A combination of all features leads to the best performance, they provide complementary information. Several ensembling techniques have been proposed recently: mixing experts (Jordan and Jacobs, 1991), model Stacking (Wolpert, 1992), Bagging and Boosting (Breiman, 1999) . We use Stacking in this task. The main reason is that other methods are relatively simple and may have large learning errors. Stacking is like an upgraded version of Bagging. The second layer of learning in Stacking is to find the right weight or the right combination.

The Stacking algorithm is divided into two layers. The first layer uses different algorithms to form n weak classifiers, and simultaneously generates a new data set of the same size as the original data set. This new data set and a new algorithm form the second layer classifier.

When using the Stacking strategy, we do not execute a simple logical processing of the weak learner, but add a layer of learner, that is, we will use the learning result of the Attention-LSTM model and the BiGRU-Capsule model as input, building an MLP model as second layer classifier, the MLP model has only one hidden layer, there are 200 hidden nodes in the layer, and a Dense layer as the output of the ensemble. The architecture of the ensemble model is shown in Figure 3.

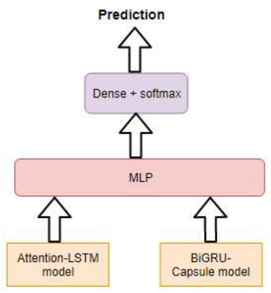


Figure 3: The architecture of the ensemble Model.

3 Experinment and Result analysis

3.1 Data Processing

The official data set is very noisy and needs to be cleaned. Preprocessing the text makes it easy for the model to extract features and representations. We perform the following preprocessing.

- Hashtags are important markers for determining sentiment or user intention. The "#" symbol is removed and the word itself is retained. e.g.: in the sentence, "#BuildTheWall and #BuildThatWall" are marked as 1 in most cases in the training data set.
- Username mentions, e.g.: words starting with "@", generally provide no information in terms of sentiment. Hence such terms are removed completely from the tweets.
- Repeated full stops, question marks and exclamation marks are replaced with a single instance with a special token "repeat" added.
- All contractions are split into two tokens by using regular expression (e.g.: "it's" is changed to "it" and "is").
- All URLs, phone numbers and date numbers are replaced respectively as "URL", "PHONENUMBER", "NUMBER".
- Emoticons (such as, ':(', ':)', ':P' and emoji etc.) are replaced as their own meanings by emotion lexicons[1].
- Tokens are converted to lower case.

3.2 Hyperparameter setting

We select the longest sentence in all cleaned data as the maximum sentence length, which is 58 characters. The processed text is then converted to word embeddings. Converting text into word embeddings represents each word of the text with a d dimensional vector (Mikolov et al., 2013). We use available pre-trained embeddings which are trained on large data set.

In the attention-LSTM model, there is mainly one LSTM layer and one attention layer. There are 300 hidden nodes in the LSTM layer. We also use the Dropout layer with rate 0.25 between the LSTM layer and the Attention layer. The purpose is to prevent over-fitting. Finally, we also use Batch normalization with a size of 0.1 behind the Attention layer, this layer is normalized for each neuron, even only need to normalize a certain neuron, rather than normalize a whole layer of neurons. The purpose is to make the model training converge faster, and the distribution of model hidden output features is more stable, which is more conducive to model learning.

In the capsule model, we build two layers of BiGRU and one layer of Capsule. In the capsule layer, our routing size is set to 5, the number of capsules is set to 10, and the size of the capsule is set to 16. For BiGRU, we set the hidden unit to 128, and a Dropout layer with size 0.25 is added between the BiGRU layer and the Capsule layer to prevent overfitting. Finally, in all models, the loss function is $binary\ crossentropy$, and the optimizer is $adam$ (Kingma and Ba, 2014).

3.3 Result analysis

For this task, we select fastText (Joulin et al., 2017), because in this task we find that the result of fastText is much better than other word vectors such as Word2vec and Glove. Table 1 is the result of different word vectors as embedding.

Word Vector	**Dim**	**macro-F1 Result**
Word2vec	300d	0.746
Glove-twitter	200d	0.763
BPEmb	300d	0.732
fastText	**300d**	**0.761**

Table 1: The result of different word vectors as embedding in the attention-LSTM model for development data set in Subtask A.

We think that the reason why fastText works better than others is that Word2vec treats each word in the corpus as an atom, and it generates a vector for each word, which ignores the internal morphological features of the word, such as: "apple" and "apples", but fastText overcomes this problem by using character-level n-grams to represent a word; fastText may have a higher dimension than Glove-twitter, indicating more features; BPemb is based on Byte-Pair Encoding, the effect of fastText is obviously better than it.

Here we compare the effects of BiGRU, LSTM and BiLSTM and find that LSTM is superior to BiGRU and BiLSTM in this model, and the results are shown in Table 2.

We compare the results achieved by our individual approaches with the submitted ensemble system in Table 3. For brevity, we only show the macro-F1 scores on the development set.

The results of our test data set and the top three results of the official rankings are shown in Table

[1]https://emojipedia.org/

Model	macro-F1 Result
Attention-BiGRU	0.751
Attention-BiLSTM	0.742
Attention-LSTM	**0.761**

Table 2: The results of using BiGRU, LSTM, BiLSTM with the attention mechanism for development data set in Subtask A.

Model	macro-F1 Result
Attention-LSTM	0.761
BiGRU-Capsule	0.758
Ensemble	**0.782**

Table 3: The result of different model for development data set in Subtask A.

4 and Table 5. From the results of our model in the test data for Subtask A, its macro-F1 is only 0.498, which is 0.284 lower than the result of the training data set at training phase of 0.782, indicating that our model may have some over-fitting.

Team	macro-F1 Result
saradhix	0.651
Panaetius	0.571
YunxiaDing	0.546
Our model	**0.493**

Table 4: The results of our test data set and the top three results of the official rankings in Subtask A.

Team	EMR Result
ninab	0.570
iqraameer133	0.568
scmhl5	0.483
Our model	**0.344**

Table 5: The results of our test data set and the top three results of the official rankings in Subtask B.

4 Conclusion

In this paper, we propose a deep learning framework to classify hate speech about immigrants and women in tweets for English. The proposed approach is based on an ensemble of attention and capsule, allowing us to explore the different directions of a neural network based methodology. Each individual approach is described in detail with a view of making our experiments replicable.

In the future, we would like to experiment with handcrafted features in addition to word-vectors and lexicon features. We would also experiment with AffectiveTweets package (Mohammad and Bravo-Marquez, 2017) such as TweetToSentiStrengthFeatureVector, TweetNLPTokenizer etc., and try to extract the NER feature to further improve the model performance.

References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Leo Breiman. 1999. Prediction games and arcing algorithms. *Neural Computation*, 11(7):1493.

Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. 2012. Detecting offensive language in social media to protect adolescent online safety. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 71–80. IEEE.

Maral Dadvar, Dolf Trieschnigg, Roeland Ordelman, and Franciska de Jong. 2013. Improving cyberbullying detection with user context. In *European Conference on Information Retrieval*, pages 693–696. Springer.

L. K Hansen and P Salamon. 2002. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Michael I. Jordan and Robert A. Jacobs. 1991. Hierarchies of adaptive experts. In *Advances in Neural Information Processing Systems*, pages 985–992.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Computer Science*.

Ritesh Kumar, Atul Kr Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *Proceedings of the*

First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018), pages 1–11.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26:3111–3119.

Saif Mohammad and Felipe Bravo-Marquez. 2017. Wassa-2017 shared task on emotion intensity. In *The Workshop on Computational Approaches To Subjectivity*, pages 34–49.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3856–3866.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Stéphan Tulkens, Lisa Hilte, Elise Lodewyckx, Ben Verhoeven, and Walter Daelemans. 2016. A dictionary-based approach to racism detection in dutch social media. *arXiv preprint arXiv:1608.08738*.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26. Association for Computational Linguistics.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Elizabeth Whittaker and Robin M Kowalski. 2015. Cyberbullying via social media. *Journal of School Violence*, 14(1):11–29.

David H Wolpert. 1992. Stacked generalization. *Neural Networks*, 5(2):241–259.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2017. Hierarchical attention networks for document classification. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

YNU_DYX at SemEval-2019 Task 5: A Stacked BiGRU Model Based on Capsule Network in Detection of Hate

Yunxia Ding, Xiaobing Zhou,[*] Xuejie Zhang
School of Information Science and Engineering
Yunnan University, Yunnan, P.R. China
`yxding01@163.com`, `zhouxb@ynu.edu.cn`, `xjzhang@ynu.edu.cn`

Abstract

This paper describes our system designed for SemEval 2019 Task 5 "Shared Task on Multilingual Detection of Hate". We only participate in subtask-A in English. To address this task, we present a stacked BiGRU model based on a capsule network system. In order to convert the tweets into corresponding vector representations and input them into the neural network, we use the fastText tools to get word representations. Then, the sentence representation is enriched by stacked Bidirectional Gated Recurrent Units (BiGRUs) and used as the input of capsule network. Our system achieves an average F_1-score of 0.546 and ranks 3rd in the subtask-A in English.

1 Introduction

Hate speech is an offensive language, a statement that a person or group attacks another person or group based on characteristics such as gender, race, religion, disability, or sexual orientation. Nockleby (Nockleby, 2000) defines hate speech as "any communication that disparages a person or a group on the basis of some characteristic such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or other characteristic." Given the huge amount of user-generated content on the Web, and in particular on social media, the problem of detecting, and therefore possibly limit the Hate Speech diffusion, is becoming fundamental, for instance for fighting against misogyny and xenophobia (Basile et al., 2019).

Microblog today has become a very popular communication tool among Internet users. Millions of users share opinions on different aspects of life everyday. And Twitter[1] is a social platform that is very popular all over the word and millions of people share their experiences, moods, attitudes toward life and discuss current issues (Pak and Paroubek, 2010). Many of the content is related to people's feelings, so many people begin to conduct emotional analysis and research on tweets. SemEval 2019 Task 5 is to detect hate speech on tweets. Task A is a binary classification task that predicts whether English or Spanish tweets for specific goals (women or immigrants) are hateful or not hateful (Basile et al., 2019). There are many studies that currently use tweets as a corpus for natural language processing (NLP). Text classification using traditional machine learning methods mainly includes Support Vector Machines (SVMs) (Gunn et al., 1998), Naive Bayes (McCallum et al., 1998) and Random Forests (Cutler et al., 2007), etc. In recent years, the use of deep neural networks for NLP has become mainstream, such as Convolutional Neural Networks (CNNs) for sentence classification (Kim, 2014) and Recurrent Neural Networks (RNNs) (Graves et al., 2013).

This task aims to predict whether the tweet for each ID is a hate speech about women or immigrants. Our system implements a stacked Bidirectional Gated Recurrent Units (BiGRUs) (Cho et al., 2014) based on a capsule network. The vector representations of words are obtained with fastTex. The result of the classification is through the output of a fully connected layer. The rest of this paper is organized as follows: Data Processing and analysis are discussed in section 2. Section 3 provides the details of the proposed model. Experiments and results are described in Section 4. Finally, we draw conclusions in Section 5.

2 Data Processing

This part describes the experimental data and data processing analysis of SemEval 2019 Task 5 subtask-A in English.

[*]Corresponding author
[1]http://twitter.com

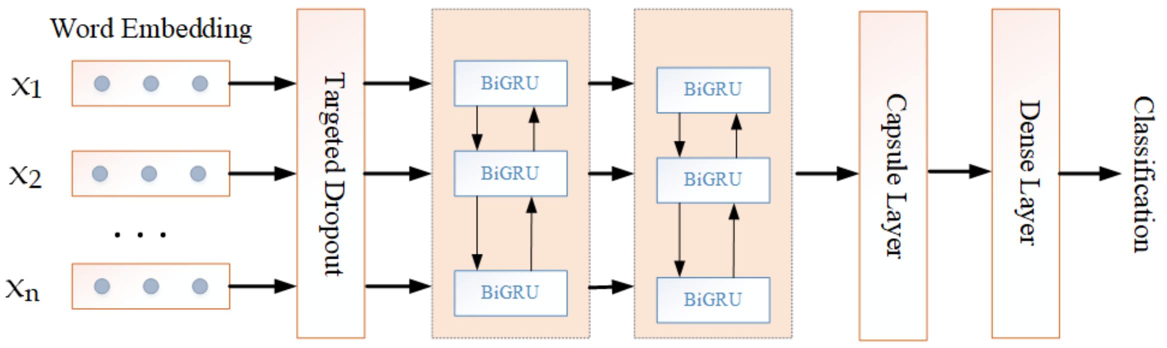


Figure 1: Neural architecture of stacked BiGRU with capsule network.

2.1 Experimental Data

This is a binary classification task of hate speech about immigrants or women. The task organizers provide training sets, development sets and test sets, respectively. Table 1 shows the data distribution of hate speech and non-hate speech in each data set. From Table 1, we can find that there are 9,000 tweets in training set, 1,000 tweets in development set and 2,971 tweets in test set.

data	hate speech	non-hate speech
training set	3,783	5,217
dev set	427	573
test set	1,252	1,719

Table 1: Distribution of labels in each datasets.

2.2 Processing Data

We perform a series of standard processing on datasets.

- All punctuation marks are removed.
- All characters are converted to lowercase.
- All hyperlinks are replaced by "url".
- All sentences are tokenized by Natural Language Toolkit (NLTK) (Bird et al., 2009).
- All numbers are replaced by "number"
- All contractions are normalized, like place "shouldn't" with "should not" and "dosen't" with "does not" and so on.
- All @specific user names are replaced with usernames, for example "@PdxPatriot1" is replaced with "username".

We consider the specific length of the sentence in the input model. If it is too long, the calculation time of the training model will increase. If it is too short, it will lose extra information. So we choose twice the average value, which is 45, as the final length of the sentence in the input model, so that the lost information will not be too much, and the calculation time will not be too long. In the training set, the development set and the test set have 473, 122, and 102 sentences respectively longer than 45, and the maximum sentence length is 65.

3 System Description

Our system can be roughly divided into two parts: the space vector representation of the words and the learning of the tweet content by the capsule network. We first map the words into a low-dimensional space vector, then feed the sentence vectors composed of these word vectors into a capsule network to learn the sentence features, and finally classify the text of the test set by a softmax function.

3.1 Word Representation

Representing a word by using a low-dimensional vector is currently the most common method in natural language processing. The fastText (Joulin et al., 2017) tool is used in our system to get the word representation of the sentences. A low-dimensional vector in fastText is associated with each word, and hidden representations can be shared between different classes of classifiers so that textual information can be used together in different classes. So fastText is a very efficient, word-based vectorization model for text classification. The pre-trained fastText embedding is used in our system[2].

3.2 Model Description

In order to enrich the word vector representation in the text, we use a stacked Bidirectional Gated Recurrent Units (BiGRUs) (Cho et al., 2014). The output of BiGRU is then used as the input to the capsule network (Sabour et al., 2017). The final result is obtained by the $softmax$ activation function in the fully connected layer. The model architecture is show in Figure 1.

Targeted Dropout Layer: *Dropout* regularization only activates some local neurons in each forward propagation, so it adds sparsity properties during training. This encourages the neural network to learn a representation that is robust to sparsification, that is, to randomly delete a set of neurons. *Targeted Dropout* (Gomez et al., 2018) sorts weights or neurons based on some measure of fast approximation weight importance and applies *Dropout* to those elements of lower importance. This approach encourages neural networks to learn more important weights or neurons. In other words, the network learns to be robust to our choice of post hoc pruning strategy. At the same time it is easy to implement with *Keras* [3].

Stacked BiGRU: To get more fine-grained sentence information, we use stacked Bidirectional Gated Recurrent Units (BiGRUs) to encode sentence information. The "stack" here refers to 2, which is 2 layers BiGRU. The information of the sentence is directional. The forward GRU can only get the information from the front to the back of the sentence, and can't encode the information from the back to the front. BiGRU better captures semantic dependencies in both directions.

Capsule Layer: The capsule network (Sabour et al., 2017) replaces a single neuron node of a traditional neural network with a neuron vector, and trains a completely new neural network in the way of Dynamic Routing, which effectively improves the low efficiency and space insensitivity of the CNN model. The capsule network is connected the same way as a fully connected network. Each capsule neuron in the previous layer is connected to each capsule neuron in the next layer. Each connection of the capsule network is also weighted. The difference is that there is a coupling coefficient on the connection of the capsule network. The coupling coefficient is determined by the iterative dynamic routing process.

[2]https://fasttext.cc/docs/en/english-vectors.html

[3]https://pypi.org/project/keras-targeted-dropout/

4 Experiments and Results

4.1 Evaluation

To evaluate the performance of the classification system, the system uses a standard evaluation metrics that includes *accuracy*, *precision*, *recall*, and F_1-*score*. In this task we use F_1-*score* to measure the performance of the proposed method. *Accuracy* is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. *Precision* is the ratio of correctly predicted positive observations to the total predicted positive observations. *Recall* is the ratio of correctly predicted positive observations to the all observations in actual class. F_1-*score* is the weighted average of *Precision* and *Recall*. *Precision* and *recall* have equal contributions to F_1-score. The formula for F_1-*score* is defined as:

$$F_1 - score = \frac{(2 * Precision * Recall)}{(Precision + Recall)} \quad (1)$$

4.2 Hyperparameter

The *Targeted Dropout* layer has two parameters, *drop_rate* and *target_rate*. In this system, these two parameters are both set to 0.55.

For the stacked BiGRU, the first layer BiGRU *units* = 64, and the second layer BiGRU *units* = 64.

The parameters of the capsule layer are set as follows: *routings* = 5, the number of caspule is 10 and the dimension is 32.

Finally, at the full connection layer output, we added two parameters, *kernel_regularizer* and *activity_regularizer*, respectively. *Kernel_regularizer* uses l_2 regularization with a parameter of 0.001, *activity_regularizer* is l_1 regularization, and the parameter is also set to 0.001.

Usually the multi-classification problem uses *categorical crossentropy* as the loss function. But our system uses *binary crossentropy* in this binary classification.

We set *epochs* = 6 and *batch size* = 64.

4.3 Experiments and Result Analysis

We conduct several experiments to gain insight into the performance of the proposed model. First we compare the normal *Dropout* and *Targeted Dropout* performance.

It can be seen from Table 2 that the performance of *Targeted Dropout* is significantly better than that of *Dropout*. Model performance increases by 5% on average F_1-*score*.

Sets	Acc	P	R	F$_1$
Dropout	0.53	0.58	**0.61**	0.52
Targeted Dropout	**0.56**	**0.64**	0.60	**0.55**

Table 2: Experimental results on test set. The values in the table are macro averages.

To determine the specific parameters of the *Targeted Dropout*, we do a lot of comparison experiments. As can be seen from Table 3, the best parameter is 0.55. This is also the parameter we submitted to the system in the competition.

Targeted Dropout	Acc	P	R	F$_1$
0.40	0.53	0.58	0.63	0.50
0.45	0.56	0.60	0.63	0.54
0.50	0.53	0.59	**0.64**	0.49
0.55	**0.56**	**0.64**	0.60	**0.55**
0.60	0.55	0.60	0.63	0.54

Table 3: Experimental results of different Targeted Dropouts on the test set.

We compare the four network architectures based on a capsule network, LSTM, GRU, BiLSTM and BiGRU. We observe that the performance of BiGRU is better than the other three in this task. Compared to MFC baseline and SVC baseline, our method increases the average F_1-*score* by 0.18 and 0.10, respectively, as is shown in Table 4.

The values of MFC baseline and SVC baseline come from the data published by the organizer[4]. To ensure the fairness of the experiment, the parameters of the capsule network remain unchanged, using the parameters mentioned in section 4.2.

5 Conclusion and Future Work

In this paper, we present a stacked BiGRU model based on a capsule network system in the task "Shared Task on Multilingual Detection of Hate". We replace *Dropout* with *Targeted Dropout*, the effect is more obvious, indicating that *Targeted Dropout* is effective in this system. At the same time, we have conducted several experiments to find the optimal parameters of *Targeted Dropout*. Through comparative experiments, BiGRU is the best model based on capsule networks.

Due to time limit, we don't tune the parameters of the capsule network. In the future, we will adjust the parameters of the capsule network to optimize the performance of the model. Secondly, we are going to try ensemble methods such as hard voting, soft voting and stacking to find the one that works best for our task. Finally, we would like to explore transfer learning technology.

[4]https://docs.google.com/spreadsheets/d/1wSFKh1hvwwQIoY8_XBVkhjxacDmwXFpkshYzLx4bw-0/edit#gid=0

Model	Acc	P	R	F$_1$
MFC baseline	**0.58**	0.29	0.5	0.37
SVC baseline	0.49	0.60	0.55	0.45
LSTM	0.55	0.60	**0.64**	0.53
GRU	0.54	0.59	0.62	0.52
BiLSTM	0.53	0.58	0.62	0.51
BiGRU	0.56	**0.64**	0.60	**0.55**

Table 4: Each model is a stacked or two-layer model, and the units in the model are all 64.

Acknowledgments

This work was supported by the Natural Science Foundations of China under Grant Nos. 61463050, 61702443 and 61762091, and the Project of Innovative Research Team of Yunnan Province under Grant No. 2018HC019.

References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

D Richard Cutler, Thomas C Edwards Jr, Karen H Beard, Adele Cutler, Kyle T Hess, Jacob Gibson, and Joshua J Lawler. 2007. Random Forests for Classification in Ecology. *Ecology*, pages 2783–2792.

Aidan N Gomez, Ivan Zhang, Kevin Swersky, Yarin Gal, and Geoffrey E Hinton. 2018. Targeted Dropout. In *International Conference on Neural Information Processing Systems*.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech Recognition with Deep Recurrent Neural Networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

Steve R Gunn et al. 1998. Support Vector Machines for Classification and Regression. *ISIS technical report*, 14(1):5–16.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Andrew McCallum, Kamal Nigam, et al. 1998. A Comparison of Event Models for Naive Bayes Text Classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer.

John T Nockleby. 2000. Hate Speech. *Encyclopedia of the American constitution*, 3(2):1277–1279.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, volume 10, pages 1320–1326.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic Routing Between Capsules. In *Advances in neural information processing systems*, pages 3856–3866.

Amrita School of Engineering - CSE at SemEval-2019 Task 6: Manipulating Attention with Temporal Convolutional Neural Network for Offense Identification and Classification

Murali Sridharan, Swapna T R
Department of Computer Science and Engineering
Amrita School of Engineering, Coimbatore,
Amrita Vishwa Vidhyapeetham, India
muralisridharan.10@gmail.com,tr_swapna@cb.amrita.edu

Abstract

With the proliferation and ubiquity of smart gadgets and smart devices, across the world, data generated by them has been growing at exponential rates, in particular social media platforms like Facebook, Twitter and Instagram have been generating voluminous data on a daily basis. According to Twitter's usage statistics, about 500 million tweets are generated each day. While the tweets reflect the users' opinions on several events across the world, there are tweets which are offensive in nature that need to be tagged under the hateful conduct policy of Twitter. Offensive tweets have to be identified, captured and processed further, for a variety of reasons, which include i) identifying offensive tweets in order to prevent violent/abusive behaviour in Twitter (or any social media for that matter), ii) creating and maintaining a history of offensive tweets for individual users (would be helpful in creating meta-data for user profile), iii) inferring the sentiment of the users on particular event/issue/topic . We **(CodaLab Team/User Name: murali_sr)** have employed neural network models which manipulate attention with Temporal Convolutional Neural Network for the three shared subtasks i) ATT-TCN (ATTention based Temporal Convolutional Neural Network) employed for shared sub-task A that yielded a best macro-F1 score of 0.46, ii) SAE-ATT-TCN(Self Attentive Embedding-ATTention based Temporal Convolutional Neural Network) employed for shared sub-task B and sub-task C that yielded best macro-F1 score of 0.61 and 0.51 respectively. Among the two variants ATT-TCN and SAE-ATT-TCN, the latter performed better.

1 Introduction

In the prevailing digital era, Deep Learning has penetrated almost all industry verticals and afforded several researchers an effective tool, in handling voluminous data and deriving meaningful inferences. Initially, (LeCun et al., 1998) invented Convolutional Neural Network (CNN) model for extraction of local features, which later proved to be the standard choice for Computer Vision tasks. (Hochreiter and Schmidhuber, 1997) the introduced LSTM (Long Short Term Memory) architecture, which went on to become the standard choice for Natural Language Processing (sequence) tasks due to the implicit ordering of the sequence data in words and sentences. Then several architectures, combining LSTM with CNN were introduced that went on to become successful for NLP tasks as well. Deep Learning techniques have leaped forward through multiple NLP tasks such as Modeling, Classification, Translation, Summarization, etc., and have proved to be better compared to traditional techniques.

Ever since social media has become ubiquitous there have been individuals who take gratuitous advantage of the anonymous nature of social media platforms, and engage themselves in rude and offensive communications. Such behaviour that prohibit free flow of communication and violate acceptable usage policy has necessitated to identify and capture the offensive posts, comments, etc., in order to prevent the dissemination of abusive behaviour in social media. (Zampieri et al., 2019b) focused on this aspect and organized a classification task with a particular focus on Twitter posts; unlike predictions of positive or negative sentiments, this task has three shared sub-tasks, intended to identify and capture the offense target as an entity. The task includes three shared sub-tasks that include:

i) Sub-Task A: Offensive language identification,
ii) Sub-Task B: Offense type categorization and

iii) Sub-Task C: Offense target identification.
i) Sub-Task A: Offensive language identification in which posts are categorized into Offensive or Not Offensive. Recently (Bai et al., 2018) empirically concluded that the association between sequence modeling and recurrent neural networks should be reconsidered and established that convolutional networks are ought to be considered for sequence modeling tasks. The TCN model can be extended followed by introduction of Attention to the output of Embedding layer and TCN layer
ii) Sub-Task B: Offense type categorization in which the Offense type is categorized into either targeted or untargeted. The objective here is to understand sentence structure by emulating the relationship between words. The sequence of words is crucial to capture the essence of sentence unlike the practice of mere focus on constituent parts of a sentence in the previous model. Based on (Lin et al., 2017), minor modifications are injected into the previous model employed in sub-task A, and introduced self-attention for embedding further to aggregate the relationship between words in a sentence and stacked attention layer, at the output of each dilated convolution blocks. **iii) Sub-Task C: Identification of target offense** in which the who, the offense is aimed at is identified and categorized into Individual, Group or Other. The same model used in the previous sub-task B is employed for this sub-task as well.

2 Related Work

In the recent past, multiple NLP tasks and papers have explored Offense identification which include (bullying, aggression, hate-speech, obscenity, insults and identity threat). (Fortuna and Nunes, 2018) has elaborately surveyed several approaches employed for automatic detection of hate speech.
(Yin et al., 2009) was one of the first to address recognition of offensive language by employing supervised classification technique along with manually developed n-gram regex matches and, contextual attributes that considered the intensity of abuse in preceding sentences. (Sood et al., 2012) indicated that certain banned words when used in appropriate manner and context, does not warrant to be categorized as abusive/offensive. Further, they showed a considerably improved scheme of profanity detection, by incorporating lists and distance metric, which enabled identification and categorization of un-normalized terms like ”@$$” or ”m0r0n”. (Chen et al., 2012) used lexical and parser features, for detecting comments from YouTube that are offensive. Without any preset semantics of toxic content, they came up with the tool that could be manipulated through a modifiable threshold. This threshold was to be treated as a measure of toxicity, filtering the online toxic content, prior to display of contents in the client's browser. Their work incorporated Support Vector Machines (SVMs) classifiers, which included regex (manually developed), n-gram, black-lists and dependency parse features, which achieved higher precision and recall values.
(Dadvar et al., 2013) affirmed that user context was crucial in the bonafide detection of cyberbullying. (Djuric et al., 2015) highlighted the effectiveness of comment embeddings in detection of hate speech, by joint modelling comments and words using Continuous-Bag of Words (C-BOW) to generate a low dimensional embedding. The embedding is passed to binary classifier for hate speech detection. (Mehdad and Tetreault, 2016) explored the significance of features to the extent of character to word and weighed the importance of each attribute. Since the style of comments, in online forums, vary from person to person, and often includes sub-standard profane English (i.e. ”f u c k e r”), learning how adjacent characters communicate with each other reveal more about the abusiveness of a comment as a whole.
(Vijayan et al., 2017) surveyed the pros and cons of several techniques of machine learning and deep learning in their comprehensive study of text classification algorithms. (Malmasi and Zampieri, 2017) employed n-gram and skip gram based SVM classifier, to detect and classify hate-speech, into three categories: Hate, Offensive and Ok. (Gambäck and Sikdar, 2017) employed multiple CNN models totaling four for Hate-Speech Classification of Twitter posts into one of the following:sexism, racism, either(sexism and racism) and not hate speech. The first model, trained on character based n-grams (4-grams), the next model trained on word vectors built using word2vec. The third model was trained on word vectors which were produced in random. The fourth model was trained on word vectors in addition to char n-gram for the classification task. The fourth model performed comparatively better in the classification task. (Waseem et al., 2017)

proposed a typology, to capture the similarity and difference between sub-tasks, and discuss their role, involved in annotating data and emulating feature construction. Their work was instrumental in identifying if the offense was targeted towards an individual or an entity, and whether the offensive language was explicit or implicit. (Zhang et al., 2018) introduced a new method, combining Convolutional Neural Network (CNN) and Gated Recurrent Unit (GRU) to perform a compartive evaluation on public datasets, and set new benchmarks, in Hate Speech Detection on Twitter.

3 Methodology and Data

Besides being fast and parallel, the important aspect of TCN is causal convolution, its capability to take any arbitrary length sequence and generate an output sequence of the same arbitrary input length.

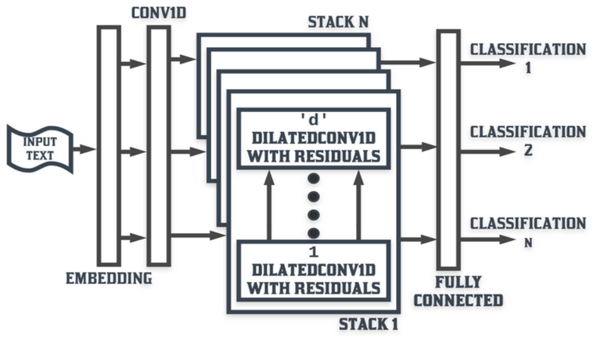


Figure 1: Temporal Convolutional Neural Network

Ever since (Bahdanau et al., 2014) introduced attention mechanism in NLP, for machine translation there have been multiple advances in memory related tasks. Further (Yin et al., 2016) established that attention based CNN performed better than attention based LSTM (Long Short Term Memory) for the answer selection task.

Here in Sub-Task A:Offense Identification, TCN was extended for sub-task A, with attention mechanism. Instead of the conventional dropout layer, a simple attention mechanism is applied at the output of embedding layer and at the output of TCN before classification layer, as illustrated in Figure 2 and Figure 3. The intention is to avoid random dropout of constituent data, which might be crucial, and introduce a mechanism with the ability, to selectively focus on input and capitalize on the crucial contributing parameters with

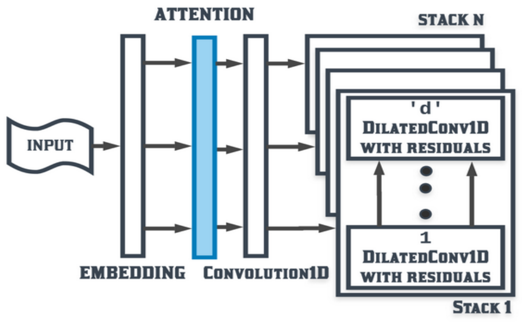


Figure 2: Temporal Convolutional Neural Network with Attention layer at the output of Embedding layer

varying attention weights and contextual vectors. ReLu (Nair and Hinton, 2010) activation is used for DilatedConvolution1D (Yu and Koltun, 2015) and softmax (Bridle, 1990) is used at the final classification layer.

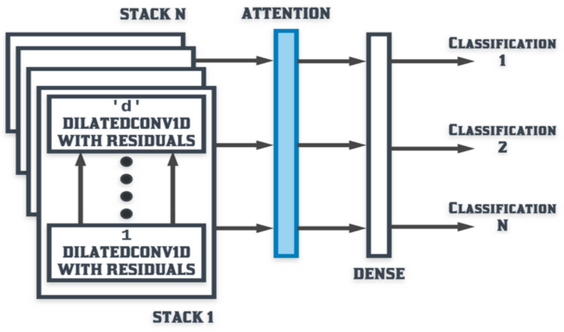


Figure 3: Temporal Convolutional Neural Network with Attention layer at the output of TCN before final Classification

Parameters	A	B	C
# Features	100*	250*	250*
# Filters	3*	5*	5*
Kernel Size	4	5	5
Dilation Range	11	11	11
Stack Count	1	1	1
Dropout Rate	0.05	0.01	0.01
Batch Size	32	32	32

Table 1: Hyper-parameters for each Sub-Task A, B and C respectively. Parameters are in numbers. *x10^2

For Sub-Task B:Automatic Categorization of Offense Type, a slight modification is introduced, to the model used for the previous sub-task, by incorporating Self Attention at the output of Embedding layer.

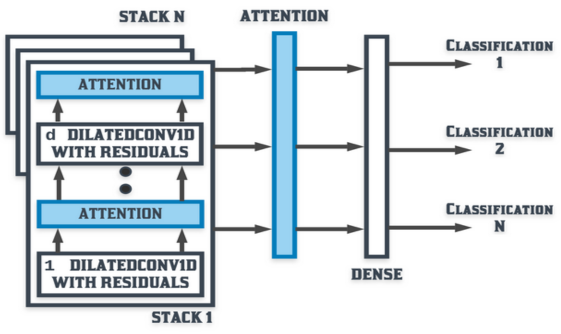


Figure 4: Temporal Convolutional Neural Network with Attention layer at the output of each Dilated-Conv1D and

Self-Attention is introduced for characterization of multiple location of the tokens, a sentence has, in addition to extraction of semantic features. Additionally, Attention layer is stacked at the output of every 'd' dilated convolution blocks, to augment the contextual vectors, as illustrated in Figure 4. For Sub-Task C:Offense Target Identification, the model used in the sub-task B was employed to identify and categorize the target of the posts into Individual (IND), Group (GRP) and Other (OTH) classes. For all the variants, binary cross-entropy loss function is employed with the focus on categorical accuracy.

The methods employed for gathering the data, preparation and compilation of dataset, used in OffensEval shared task is described in Zampieri et al. (2019a). Two additional datasets, Kaggle Toxic Comment Classification dataset and TRAC-1 Aggression Identification in Social Media Shared Task dataset were used for sub-task A and sub-task B respectively.

In 2018, Kaggle hosted a Toxic Comment Classification competition in association with Jigsaw, which focused on classifying Wikipedia comments into one of six categories: insult, obscene, severe toxic, threat & identity hate and toxic. The instances which do not fall into one of the six categories are clean. All the six toxic categories are mapped to Offensive (OFF) class and the clean instances are mapped to Not Offensive (NOT) class.

The mapped instances were combined with the training dataset, provided for sub-task A, which produced a total of 172811 instances, of which 20625 instances were Offensive and 152186 in-

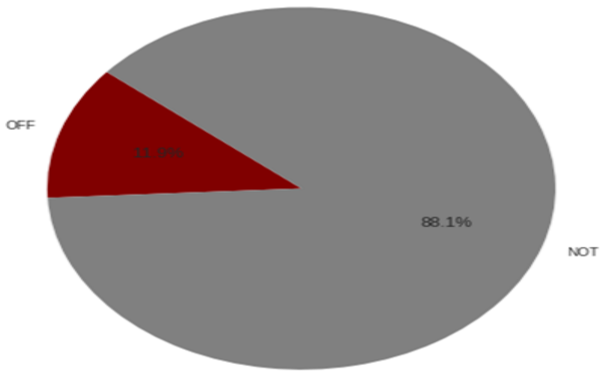


Figure 5: Shared Sub-Task A, training data instance share (OFF and NOT)

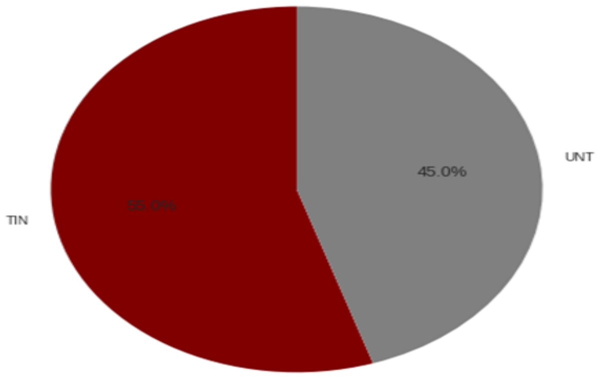


Figure 6: Shared Sub-Task B, training data instance share (TIN and UNT)

stances were clean, as depicted in Figure 5. For the training data of Sub-Task B, TRAC-1 data (Kumar et al., 2018) was used, in addition to the provided training data, producing a total of 14174 instances containing 7799 Targeted Insults and Threats (TIN), and 6375 Untargeted (UNT) instances. No other additional training dataset was used, apart from the provided dataset for Sub-Task C. It comprised of 3876 Offensive instances, of which 1074 Offensive instances belong to Individual (IND) category, 2407 Offensive instances belong to Group (GRP) category and 395 Offensive instances target belong to (OTH) category.

4 Results

The macro averaged F1 was employed as the official metric for all the sub-tasks involved in this task accounting for the high class imbalance ratio. Our first model (ATT-TCN), employed for the shared Sub-Task A, produced an overall accuracy of 65.81% (best of 3 for evaluation test data @CodaLab). The second variant (SAE-ATT-TCN), employed for the shared Sub-Task

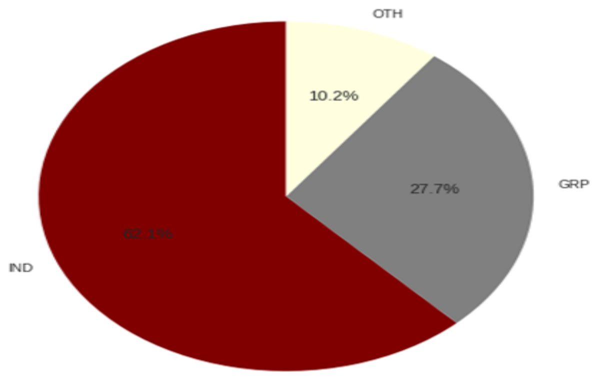


Figure 7: Shared Sub-Task C, training data instance share (IND, GRP and OTH)

B and Sub-Task C produced an overall accuracy of 75.83% and 61.5% (best of 3 for Evaluation Test data @CodaLab), respectively. The neural network model generation, fine-tuning and the evaluation test data prediction, all the activities have been executed in Google Colaboratory environment, utilizing the on hand GPU hardware accelerator. The cross validation results, and the detailed evaluation test data results have been listed in the tables accordingly.

System	Accuracy
Base ATT-TCN	0.9477
SAE-ATT-TCN[1]	0.7144
SAE-ATT-TCN[2]	0.7215

Table 2: Cross-Validation Results for Sub-Tasks A,[1]B and [2]C respectively.

System	F1 (macro)	Accuracy
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
Base ATT-TCN	**0.4682**	**0.6581**

Table 3: CodaLab Test Results for Sub-Task A.

System	F1 (macro)	Accuracy
All TIN baseline	0.4702	0.8875
All UNT baseline	0.1011	0.1125
SAE-ATT-TCN	**0.6164**	**0.7583**

Table 4: CodaLab Test Results for Sub-Task B.

The confusion matrices for the best performing variant of each Sub-Task have been depicted in Table 6, Table 7 & Table 8. In Table 6, for Sub-Task A, it is evident that the number of NOT Offensive instances (540) have been predicted correctly attributing to the higher count of training data instances for that class, and count of correct Offensive (OFF) instances prediction is less attributing to the less training instances for that category. The higher number of false positives for the Sub-Task A clearly indicate not so good classification performance of the variant ATT-TCN. The higher number of true positives for Targeted Insult and Threat (164 TIN instances), and the lesser true negatives (18 UNT instances), in Table 7, indicate the model has better generalization ability, and performed significantly better, compared to the previous model.

In Table 8, it is clear that the Group (GRP) offense target classification is predicted correctly compared to Other (OTH), and Individual (IND) categories respectively.

System	F1 (macro)	Accuracy
All GRP baseline	0.1787	0.3662
All IND baseline	0.2130	0.4695
All OTH baseline	0.0941	0.1643
SAE-ATT-TCN	**0.5132**	**0.615**

Table 5: CodaLab Test Results for Sub-Task C.

	NOT	OFF
NOT	540	80
OFF	214	26

Table 6: Sub-Task A, Confusion Matrix for Base ATT-TCN at threshold 0.45

	TIN	UNT
TIN	164	49
UNT	9	18

Table 7: Sub-Task B, Confusion Matrix for SAE-ATT-TCN at threshold 0.70

	GRP	IND	OTH
GRP	44	27	7
IND	10	81	9
OTH	14	15	6

Table 8: Sub-Task C, Confusion Matrix for SAE-ATT-TCN at threshold 0.55

5 Conclusion

Based on the results, it is evident that SAE-ATT-TCN has performed significantly better than the base model ATT-TCN. From Sub-Task A, we learned that rather than going ahead with random sampling for train and test split, proceeding with a categorical split, to the best possible even ratio, would increase the generalization ability. When the dataset has class imbalance ratio, retaining even number of instances for each class as much as possible, would ensure normal distribution of the instances for each class. Such data distribution would not be skewed for a particular class which ensure better generalization capability leading to improved classification accuracy. We note that processing the sequence in both the directions (forward and backward) would further improve the classification performance attributing to better context and semantic representation learning capabilities. We are working on Bi-Directional Attention based Temporal Convolutional Network model. Our participation in the SemEval 2019: Task 6 competition has been a very good learning experience for our team, and we are eager to learn from other best performing entries.

Acknowledgement

Our team would like thank and appreciate the SemEval 2019 organizing team for affording an opportunity to participate in such classification task. We look forward to additional NLP classification tasks.

References

Dzmitry Bahdanau, Yoshua Bengio, and Kyunghyun Cho. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

John S. Bridle. 1990. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*.

Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. 2012. Detecting offensive language in social media to protect adolescent online safety. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Conference on Social Computing*, pages 71–80. IEEE.

Maral Dadvar, Franciska de Jong, Roeland Ordelman, and Dolf Trieschnigg. 2013. Improving cyberbullying detection with user context. In *Advances in Information Retrieval*, pages 693–696. Springer.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pages 29–30. International World Wide Web Conferences Steering Committee.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ritesh Kumar, Shervin Malmasi, Atul Kr. Ojha, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC)*, Santa Fe, USA.

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Zhouhan Lin, Mo Yu, Cicero Nogueira dos Santos, Minwei Feng, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 467–472.

Yashar Mehdad and Joel Tetreault. 2016. Do characters abuse more than words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*.

Sara Owsley Sood, Elizabeth Churchill, and Judd Antin. 2012. Using crowdsourcing to improve profanity detection. In *2012 AAAI Spring Symposium Series*.

V. K. Vijayan, K. R. Bindu, and L. Parameswaran. 2017. A comprehensive study of text classification algorithms. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1109–1113.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Dawei Yin, Brian D Davison, April Kontostathis, Zhenzhen Xue, Liangjie Hong, and Lynne Edwards. 2009. Detection of harassment on web 2.0. *Proceedings of the Content Analysis in the WEB*, 2:1–7.

Wenpeng Yin, Bowen Zhou, Bing Xiang, and Hinrich Schütze. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.

Fisher Yu and Vladlen Koltun. 2015. Multi-scale context aggregation by dilated convolutions.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ziqi Zhang, Jonathan Tepper, and David Robinson. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

bhanodaig at SemEval-2019 Task 6: Categorizing Offensive Language in social media

Ritesh Kumar
Department of CSE
IIT(ISM) Dhanbad
India, 826004
ritesh4rmrvs@gmail.com

Guggilla Bhanodai
Department of CSE
IIT(ISM) Dhanbad
India, 826004
bhanodaig@gmail.com

Rajendra Pamula
Department of CSE
IIT(ISM) Dhanbad
India, 826004
rajendra@iitism.ac.in

M. R. Chennuru
Department of CSE
IIT(ISM) Dhanbad
India, 826004
cmr.mahesh@gmail.com

Abstract

This paper describes the work that our team bhanodaig did at Indian Institute of Technology (ISM) towards OffensEval i.e. identifying and categorizing offensive language in social media. Out of three sub-tasks, we have participated in sub-task B: automatic categorization of offensive types. We perform the task of categorizing offensive language, whether the tweet is targeted insult or untargeted. We use Linear Support Vector Machine for classification. The official ranking metric is macro-averaged F1. Our system gets the score 0.5282 with accuracy 0.8792. However, as new entrant to the field, our scores are encouraging enough to work for better results in future.

1 Introduction

Social media has become most popular among users in these days. Based on survey (Johnson et al., 2011), it has been observed that 70% of teenagers use social media sites on daily basis. Users share their views with help of social media like twitter, facebook, instagram, youtube. Ritesh et al. (Kumar et al., 2018a) tried to identify hate speech. On the one hand Users get benefited from social media by learning or interacting with other users on the other hand they face offensive online contents. With exponential growth of social media it has become quite significant to identify and categorize offensive language in social media.

A key challenge among researchers is to automatically categorization of offense type languages in social media. few research have been performed but it is still a hot topic among researchers. keeping it in mind, we develop a system that could categorize offensive language in social media. The relevant shared task description, data and results are described in the paper (Zampieri et al., 2019b).

In this paper, we use Linear Support Vector Machine (LSVM) for classifying and identifying offensive language in social media. We use snowball stemmer to find out root words. Also, we have used unigram and bigram language models without stopwords.

The rest of the paper is organized as follow. Section 2 describes related work. The proposed methodology and used data is described in section 3. Section 4 describes results obtained after experiment. Finally, we conclude and future work in section 5.

2 Related Work

The interest in identifying and categorizing aggression, cyber-bullying and hate speech, particularly on social media, has been growing in recent years. This topic has attracted attention from researchers interested in linguistic and sociological features of aggression, and from engineers interested in developing tools to deal with aggression on social media platforms. In this section, we review a number of studies and briefly discuss their findings. For a recent and more comprehensive survey on hate speech detection we recommend (Schmidt and Wiegand, 2017) and (Fortuna and Nunes, 2018).

Davidson et al. (Davidson et al., 2017) used crowd source to label a sample of tweets into three categories: hate speech, only offensive and those with neither. The Hate Speech Detection dataset used in (Malmasi and Zampieri, 2017) and a few other recent papers such as (ElSherief et al., 2018; Gambäck and Sikdar, 2017; Zhang et al., 2018).

A proposal of typology of abusive language sub-tasks is presented in (Waseem et al., 2017). For studies on languages other than English has been described in (Su et al., 2017) on Chinese and (Fišer et al., 2017) on Slovene. Finally, for recent discussion on identifying profanity vs. hate speech is discussed in (Malmasi and Zampieri, 2018). This work highlighted the challenges of distinguishing between profanity, and threatening language which may not actually contain profane

System	F1 (macro)	Accuracy
All TIN baseline	0.4702	0.8875
All UNT baseline	0.1011	0.1125
LSVM	**0.5282**	**0.8792**

Table 1: Results for Sub-task B using model LSVM and best result is highlighted with boldface.

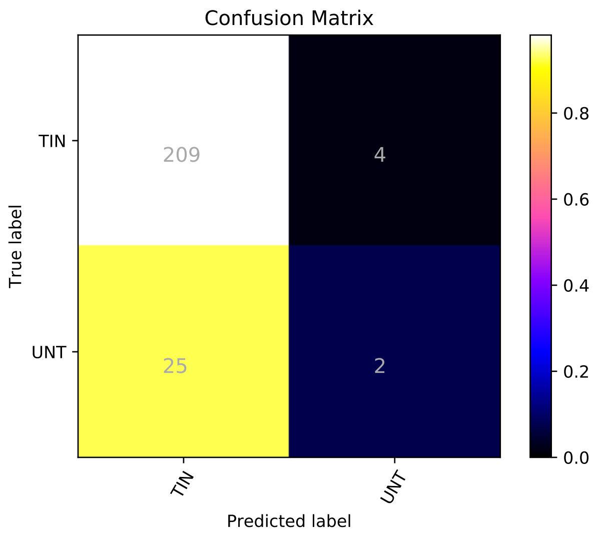


Figure 1: Sub-task B, LSVM

language.

Additionally, the related work has been performed in the related workshops such as TA-COS[1], Abusive Language Online[2], and TRAC[3] and related shared tasks such as GermEval (Wiegand et al., 2018) and TRAC (Kumar et al., 2018b).

3 Methodology and Data

The description of our system and different runs has been described in this section. We have been provided training dataset with 13,240 tweets, a trial set with 320, and a test set with 860 (Zampieri et al., 2019a). Each instance is composed of a tweet and its respective labels for tasks A, B and C. The three levels/subtasks are as follows:
Task A : Whether the tweet is offensive (OFF) or non-offensive (NOT).
Task B : Whether the tweet is targeted (TIN) or untargeted (UNT).
Task C : If the target is an individual(IND), group (GRP) or other (OTH; e.g., an issue or an organisation).

We have focused on subtask B. In our methodology, tweets are preprocessed by replacing following words with corresponding words shown below:

what's	$\rightarrow$ what is
've	$\rightarrow$ have
can't	$\rightarrow$ can not
n't	$\rightarrow$ not
i'm	$\rightarrow$ i am
're	$\rightarrow$ are
'd	$\rightarrow$ would
'll	$\rightarrow$ will
'scuse	$\rightarrow$ excuse

followed by stemming words with snowball stemmer. LSVM is used for classification. We perform the task for only categorizing offensive language in social media, whether tweet is targeted insult or untargeted. SVM is categorizing offensive language in social media. For this, tf-idf of words unigrams and bigrams (without stopwords) that are occured at least 3 times are considered as features with 12 normalization.

[1]http://ta-cos.org/
[2]https://sites.google.com/site/abusivelanguageworkshop2017/
[3]https://sites.google.com/view/trac1/home

4 Results

In this section, we describe our experimental results. The official ranking metric is macro-averaged F1. We have included accuracy here as well for comparison. In table 1 , we see that we get the best result **0.5282** with accuracy **0.8792** using LSVM model. All TIN (targeted insult) baseline has got the score 0.4702 with accuracy 0.8875 and All untargeted baseline has got the score 0.1011 with accuracy 0.1125. The confusion matrix has been shown in figure 1.

5 Conclusion and future work

This year we participated in OffensEval sub-task B i.e. automatic categorizing offensive language in social media. We use LSVM model for classification. While there can be no denial of the fact that our overall performance is average, initial results are suggestive as to what should be done next. As we have taken ngrams for training set, our model unable to handle OOV (out of vocabulary) words. Pretrained word embeddings would have handled this problem. LSTM with these word embeddings might give better results. We explore these models in coming future.

References

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Darja Fišer, Tomaž Erjavec, and Nikola Ljubešić. 2017. Legal Framework, Dataset and Annotation Schema for Socially Unacceptable On-line Discourse Practices in Slovene. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Timothy Johnson, Robert Shapiro, and R Tourangeau. 2011. National survey of american attitudes on substance abuse xvi: Teens and parents. *The National Center on Addiction and Substance Abuse*, 2011.

Ritesh Kumar, Guggilla Bhanodai, Rajendra Pamula, and Maheshwar Reddy Chennuru. 2018a. Trac-1 shared task on aggression identification: Iit (ism) @ coling18. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 58–65.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018b. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 467–472.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Huei-Po Su, Chen-Jie Huang, Hao-Tsung Chang, and Chuan-Jie Lin. 2017. Rephrasing Profanity in Chinese Text. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

BNU-HKBU UIC NLP Team 2 at SemEval-2019 Task 6: Detecting Offensive Language Using BERT model

Zhenghao Wu **Hao Zheng** **Jianming Wang** **Weifeng Su** **Jefferson Fong**

{1630003054,1630003067,1630003049}@mail.uic.edu.hk
{wfsu,jeffersonfong}@uic.edu.hk

Computer Science and Technology, Division of Science and Technology
BNU-HKBU United International College
Zhuhai, Guangdong, China

Abstract

In this study we deal with the problem of identifying and categorizing offensive language in social media. Our group, BNU-HKBU UIC NLP Team2, use supervised classification along with multiple version of data generated by different ways of pre-processing the data. We then use the state-of-the-art model Bidirectional Encoder Representations from Transformers, or BERT (Devlin et al. (2018)), to capture linguistic, syntactic and semantic features. Long range dependencies between each part of a sentence can be captured by BERT's bidirectional encoder representations. Our results show 85.12% accuracy and 80.57% F1 scores in Subtask A (offensive language identification), 87.92% accuracy and 50% F1 scores in Subtask B (categorization of offense types), and 69.95% accuracy and 50.47% F1 score in Subtask C (offense target identification). Analysis of the results shows that distinguishing between targeted and untargeted offensive language is not a simple task. More work needs to be done on the unbalance data problem in Subtasks B and C. Some future work is also discussed.

1 Introduction

Social media is an essential part of human communication today. People can share their opinions in this platform with anonymity. Some people use offensive language and hate speech casually and frequently without taking any responsibility for their behavior. For this reason, SemEval 2019 (Zampieri et al. (2019b)) set up the task OffensEval: identifying and categorizing offensive language in social media. This task is divided into three subtasks: offensive language identification, automatic categorization of offensive types, and offence target identification.

Our group uses the Natural Language Processing (NLP) latest model, Bidirectional Encoder Representations from Transformers (BERT). It is a general-purpose "language understanding" model trained on a large text corpus such as Wikipedia (Devlin et al. (2018)). After fine-tuning, the model can be used for downstream NLP tasks. Because BERT is very complex and is the state-of-art model, it is prudent for us not to change its internal structure. Hence, we focus on preprocessing the data and error analysis. After much experimentation with the data, such as translating emoji into words, putting more weight on some metaphorical words, removing the hashtag and so on, we find that using the original data will give the best performance. The reason for this is perhaps if we remove some information from the sentence, some features that affect the prediction result will be lost. So we end up using the original data to train our model.

2 Related Work

Much research has been done in detecting offensive language, aggression, and hate speech in user-generated content. In recent years, researches tend to follow several approaches: use a simple model with logistic regression to perform detection, use a neural network model, or use some other methods.

For the simple model, Davidson and Warmsley (Davidson et al. (2017)) used a sentiment lexicon designed for social media to assign sentiment scores to each tweet. This is an effective way to identify potentially offensive terms. Then they use logistic regression with $L2$ regularization to detect hate speech in social network.

Neural network models use n-gram, skip-gram or some other methods to extract features from the data. These features are used to train different models. The results produced by these models will be used as the input for training the meta-classifier (e.g. Malmasi and Zampieri (2018))

For other methods, using bag-of-words is an effective way to detect hate speech, but it is difficult to distinguish hate speech from text with offensive words that are not hate speech (Kwok and Wang (2013)). For identifying the targets and intensity of hate speech, syntactic features method is a good method (Burnap and Williams (2015)).

3 Methodology and Data

Only the training data provided by the organizer (Zampieri et al. (2019a)) are used in training our model. The data contain 13,240 pieces of tweet that had been desensitized (replacing the user names and website URLs). There are three labels that are labeled with crowdsourcing for each of the three subtasks. Gold labels obtained through crowdsourcing are confirmed by three annotators. We segmented the training set by 90% for the training set, 5% for the cross-validation set, and 5% for the test set.

Because some offensive language is subtle, less ham-fisted, and sometimes cross sentence boundary, the model trained for this task must make full use of the whole sentence content in order to extract useful linguistic, syntactic and semantic features which may help to make a deeper understanding of the sentences, while at the same time less subjected by the noisiness of speech. So, we use BERT in all three subtasks. Unlike most of the other methods, BERT uses bidirectional representation to make use of the left and right context to gain a deeper understanding of a sentence by capturing long range dependencies between each part of the sentence.

The uncased base version of the pre-trained model files [1] is used during the entire training. The training data are processed in many ways to fine-tune the model. Processing methods include removing all username tags, URL tags and symbols, converting all text to lowercase, and translating emoji into text[2]. One or more of the above methods is selected to process the training data, and then use the processed data to train the model.

In Subtask A, the accuracy after the various operations is shown in the following table.

Preprocessing	**Accuracy**
Original Data	0.8184
Remove tag & symbols	0.8126
Emoji translation v1	0.8081
Emoji translation v2	0.7960

Table 1: Training results for Sub-task A.

After all attempts, the best performing model for Subtask A is the model trained by the original data. Therefore, the original data are also used in the training of the Subtasks B and C models.

4 Results

For Subtask A, The BERT-Base, Uncased, original training data model get macro F1 score of 0.8057 and total accuracy of 0.8512.

For Subtask B, The BERT-Base, Uncased, original training data model get macro F1 score of 0.50 and total accuracy of 0.8792.

For Subtask C, The BERT-Base, Uncased, original training data model get macro F1 score of 0.5047 and total accuracy of 0.6995.

Results table and confusion matrices for Subtasks A, B and C are shown below.

[1] BERT-Base, Uncased: `https://storage.googleapis.com/bert_models/2018_10_18/uncased_L-12_H-768_A-12.zip`

[2] In the process of translating emoji characters, v1 and v2 methods were used. **v1**: Translate all emoji characters into official character name listed in the Unicode®11.0.0 Standard. **v2**: In addition to "v1" of processing of all emoji characters, the selected 97 emotional emoji characters are translated into manually determined emotional words.

System	F1 (macro)	Accuracy
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
BERT-Base, Uncased, original training data	**0.8057**	**0.8512**

Table 2: Results for Sub-task A.

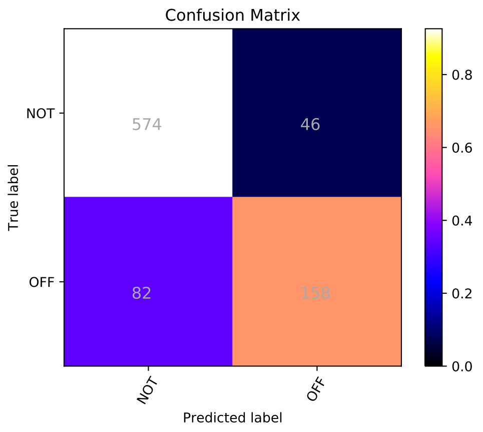


Figure 1: Sub-task A, BNU-HKBU UIC NLP Team 2 CodaLab 527070 BERT-Base, Uncased, original training data

System	F1 (macro)	Accuracy
All TIN baseline	0.4702	0.8875
All UNT baseline	0.1011	0.1125
BERT-Base, Uncased, original training data, 0.5 threshold	**0.5000**	**0.8792**
BERT-Base, Uncased, orginal training data, 0.65 threshold	0.4702	0.8875

Table 3: Results for Sub-task B.

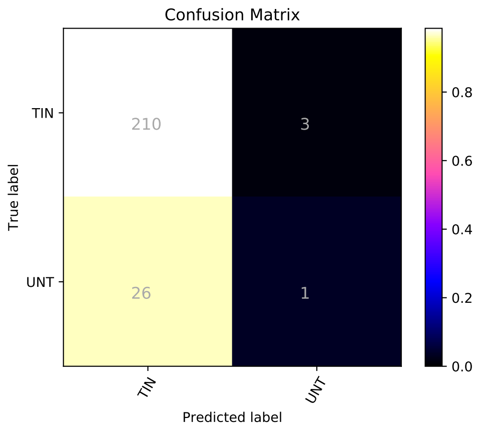


Figure 2: Sub-task B, BNU-HKBU UIC NLP Team 2 CodaLab 531958 BERT-Base, Uncased, original training data

System	F1 (macro)	Accuracy
All GRP baseline	0.1787	0.3662
All IND baseline	0.2130	0.4695
All OTH baseline	0.0941	0.1643
BERT-Base, Uncased, original training data	**0.5047**	**0.6995**

Table 4: Results for Sub-task C.

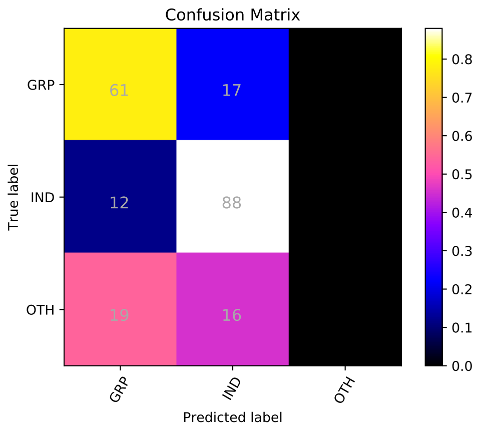


Figure 3: Sub-task C, BNU-HKBU UIC NLP Team 2 CodaLab 535873 BERT-Base, Uncased, original training data

5 Conclusions

Our model for Subtask A ranked 6 out of the 103 groups. This shows that BERT can identify offensive language. However, the results for Subtasks B and C are not as good. We will try to explain the possible reasons using error analysis.

First, some speech may be miss-labeled by annotators. In our test set, some predictions were judged as wrong, but our manual examination shows the predictions seem correct. For example, *"@USER B***hhhhh I'm jell"* and *"@USER Crazy Russian dude owns all your data"* are both labeled **NOT** (not offensive). The model, as well as our manual examination, deem these as offensive.

Second, we also notice a problem is that it is hard for our model to understanding some specific noun such as people name when our training data is not enough. For example, our model predict sentence *"Hitler will be so proud of David Hogg"* as not offensive. The word "Hitler" has a very special meaning that can makes an otherwise innocent sentence to be offensive. Our model presently can't detect this.

Another problem is emoji characters in offensive languages, which usually contains strong emotions. And may be used to express irony. So emoji characters are translated by two methods[2] to help BERT model understand the meaning of tweet posts. But the results show that both translation methods lead to a drop in accuracy. The main reason should be that some emoji characters contain different meanings in different contexts. For example, 🙂 (Slightly Smiling Face) can contain emotion of happy but also banter as well. Thus, it is difficult to understand the meaning of emoji characters in context.

Moreover, unbalanced data is a big problem. In Subtask B, few sentences are predicted as untargeted, and in Subtask C, no sentence is predicted as in the Others category. This leads to a low F1 score in these subtasks. Over-sampling in less numerous categories would not work not well in our task, and threshold moving only slightly raises the F1 score. To deal with this problem as future work, we may have to remove the labels and use unsupervised learning.

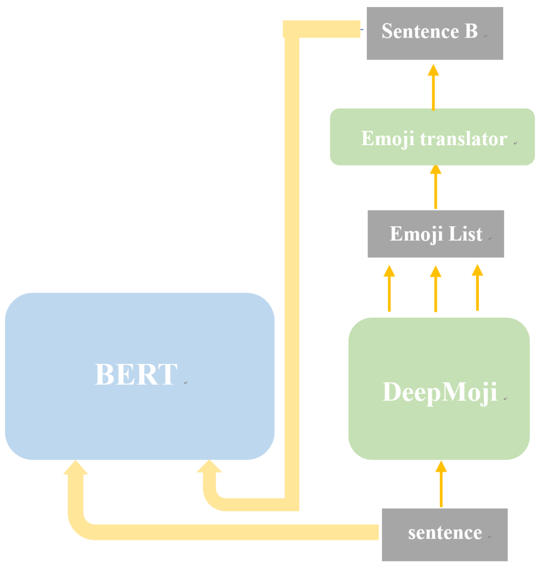


Figure 4

For future work, we notice that offensive languages often contain strong emotions such as angry, banter or taunt. This emotion and other useful contents may be improved by using DeepMoji (Felbo et al. (2017)), which translates a sentence into an emoji list to express a sentence's hidden information, such as sentiment and sarcasm. A list of emoji related to the meaning of a sentence produced by DeepMoji can be used to help BERT to better classify the sentence categories, as show in the Figure 4. The last step is to put the original sentence and the encoded new sentence as input for BERT's sentence-pair classification task.

References

Pete Burnap and Matthew L. Williams. 2015. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making.

Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *ICWSM*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524*.

Irene Kwok and Yuzhou Wang. 2013. Locate the hate: Detecting tweets against blacks. In *AAAI*.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in discriminating profanity from hate speech. *J. Exp. Theor. Artif. Intell.*, 30:187–202.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

CAMsterdam at SemEval-2019 Task 6: Neural and graph-based feature extraction for the identification of offensive tweets

Guy Aglionby†, Christopher Davis†, Pushkar Mishra‡, Andrew Caines†, Helen Yannakoudakis†, Marek Rei†, Ekaterina Shutova* & Paula Buttery†
† Department of Computer Science & Technology, University of Cambridge, U.K.
{ga384,ccd38,apc38,hy260,mr472,pjb48}@cam.ac.uk
‡ Facebook AI, London, U.K.
pushkarmishra@fb.com
* Institute for Logic, Language and Computation, University of Amsterdam, Netherlands
e.shutova@uva.nl

Abstract

We describe the CAMsterdam team entry to the SemEval-2019 Shared Task 6 on offensive language identification in Twitter data. Our proposed model learns to extract textual features using a multi-layer recurrent network, and then performs text classification using gradient-boosted decision trees (GBDT). A self-attention architecture enables the model to focus on the most relevant areas in the text. We additionally learn globally optimised embeddings for hashtags using node2vec, which are given as additional tweet features to the GBDT classifier. Our best model obtains 78.79% macro F1-score on detecting offensive language (subtask A), 66.32% on categorising offence types (targeted/untargeted; subtask B), and 55.36% on identifying the target of offence (subtask C).

1 Introduction

The SemEval-2019 shared task 6 ('OffensEval') involved three sub-parts: the classification of tweets as offensive or not (subtask A), classifying whether they are targeted insults or not (subtask B), and finally whether the targeted insults are aimed at an individual, group or otherwise (subtask C). Further details may be found in the shared task report (Zampieri et al., 2019b). Here we describe CAMsterdam's competition entry.

In recent years, there has been a growing interest in the automatic detection of offensive opinions expressed in online texts, including those posted in discussion forums, news article comment sections, and social networks. Such detection is not straightforwardly a matter of identifying texts containing obscene words (Malmasi and Zampieri, 2018); offensiveness often arises from the context, current affairs, world knowledge, the use of acronyms and slang, and the identity of the authors and audience. Therefore the task is a challenging one, but one with real world impact: if measures can be taken to identify and curtail trolling, the toxicity of the internet can to some extent be reduced. There is evidence that online harassment is connected with oppression, violence and suicide (Dinakar et al., 2011; Sood et al., 2012; Wulczyn et al., 2017), and there may moreover be reasons for concern about the perpetrator's wellbeing along with that of the victims (Cheng et al., 2017).

Our approach to the task extends the work of Mishra et al. (2018b), who extract features from tweets using an RNN for subsequent use in a gradient-boosted decision tree (GBDT) (Ke et al., 2017). Firstly, we experiment with changes to the RNN, including the use of self-attention (Rei and Søgaard, 2019) and ELMo embeddings (Peters et al., 2018). Secondly, we add additional features to the GBDT, including globally-optimised hashtag embeddings learned from a graph of tweet contents using node2vec (Grover and Leskovec, 2016). We show that this method of learning distributional information about hashtags improves performance over just learning their embeddings within a RNN.

2 Related Work

There has been much work characterising offensive online discourse including hate speech and cyberbullying (Warner and Hirschberg, 2012; Kwok and Wang, 2013; Xu et al., 2013; Waseem et al., 2017; Ribeiro et al., 2018). This work also includes creating datasets for training and evaluating detection models, for example the Hate Speech Twitter Annotations and Wikipedia Comments Corpora (Waseem and Hovy, 2016; Davidson et al., 2017; Wulczyn et al., 2017). Most work has been conducted on English data – tweets in particular – with some extensions to other domains (e.g. hacking forums (Caines et al., 2018)) and other languages (e.g. Arabic (Mubarak et al., 2017), Chinese (Su et al., 2017), Slovene (Fišer et al., 2017)).

Automated detection approaches have drawn on traditional document classification methods for spam detection and sentiment analysis, and tend to use lexical and syntactic features (Nobata et al., 2016; Li et al., 2017; Bourgonje et al., 2018). Machine learning techniques range from logistic regression (Cheng et al., 2015) to support vector machines (Yin et al., 2009) to neural networks (Gambäck and Sikdar, 2017).

We draw on the work by Mishra and colleagues, who used a character-based recurrent neural network to form contextual word representations of out-of-vocabulary words (Mishra et al., 2018b), and moreover employed graph-based author embeddings to represent group behaviour within social networks, significantly improving abuse detection (Mishra et al., 2018a). In this shared task, we do not have access to author information, but instead adapt the approach by building a graph of the tokens which occur in the training data, a method described in further detail in Section 4.4.

3 Data

The OffensEval shared task uses the Offensive Language Identification Dataset (OLID) (Zampieri et al., 2019a), which hierarchically labels tweets according to whether or not they are offensive, whether any offence is targeted, and if so targeted at whom: an individual, a group or otherwise. The three subtasks in this shared task correspond to predicting labels at each level of granularity. The data is structured to allow this: all tweets presented in subtask B are guaranteed to be offensive, and all of those in subtask C are targeted.

Tweets were collected by using the Twitter API to search for terms that are frequently associated with offensive behaviour. These included political keywords, as political content may attract a disproportionate amount of offensive comments. The dataset is evenly split between tweets sourced from these keywords and non-political ones. The authors additionally found that an effective strategy for gathering offensive tweets was to search for those flagged by Twitter's safe search feature. All tweets were anonymised by replacing usernames and URLs with placeholder tokens.

Each of the $14,100$ collected tweets were manually annotated by at least two annotators; where the original two annotators disagreed on a tweet, it was further annotated until agreement reached 66%. Table 1 presents the number of tweets in each category.

A	**B**	**C**	**Train**	**Test**	**Total**
OFF	TIN	IND	2,407	100	2,507
OFF	TIN	OTH	395	35	430
OFF	TIN	GRP	1,074	78	1,152
OFF	UNT	—	524	27	551
NOT	—	—	8,840	620	9,460
All			13,240	860	14,100

Table 1: Count of tweets in each category of OLID (Zampieri et al., 2019a).

4 Methodology

In this section, we extend the model proposed by Mishra et al. (2018b) for offensive language classification. The architecture uses a 2-layer RNN, optimised using Adam (Kingma and Ba, 2015), to predict the class of a given tweet. The pre-softmax activation values from the output layer are given as input to a GBDT for final classification. Using the GBDT for classification was found to give better results compared with predicting from the RNN directly, and allows us to include additional features into the model. The RNN is initialised with pre-trained word embeddings which are fine-tuned during training. For previously unseen words, we follow Mishra et al. (2018b) in using a neural character-based compositional model to generate plausible embeddings of unseen words. This component is optimised to compose context-aware character embeddings into word-level embeddings that are similar to the pre-trained representations, trained on words for which the embeddings are available. This methodology is effective in generating reasonable quality embeddings in instances where words were deliberately obscured to evade detection.

Following common practice in named entity recognition (Sang and De Meulder, 2003), where fine-grained labels are used to improve performance on the sequence labeling task, we take advantage of the hierarchical labels available for each tweet. For subtasks A and B we train a model to predict all cascading labels, and sum the probabilities of labels under the relevant class to make a final prediction. For example, for subtask A the model is trained to predict between 5 classes: not-offensive (NOT), offensive but not targeted (UNT), targeted towards an individual (IND), towards a group (GRP), and towards any other target (OTH). We classify a tweet as offensive if the cumulative probability mass for UNT, IND, GRP, and OTH is greater than NOT.

We also introduce several architectural extensions to the Mishra et al. (2018b) model. Firstly, we augment the core RNN with ELMo embeddings and a self-attention mechanism. Secondly, we add both the post-softmax output from the RNN as well as graph-based representations of tweets as input features to the GBDT classifier. We provide details of each extension in the following sections.

For each subtask, we experiment with combinations of the above and additionally tune the RNN type (between LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014)), dimension, and batch size, whether to use character n-grams ($n \in [1, 4]$), and, when used, the size of self-attention layers. We also run experiments using the unmodified model to find which pre-trained embeddings give the best performance. We compare publicly available embeddings trained using Word2Vec (Mikolov et al., 2013), FastText (Mikolov et al., 2018), and GLoVe (Pennington et al., 2014).

4.1 ELMo

We use embeddings generated from ELMo concatenated with pre-trained word embeddings as input to the RNN. ELMo generates embeddings on a character level, so does not share the same out-of-vocabulary issue as pre-trained embeddings and is always able to generate a word representation. We used the largest pre-trained model available online[1], and learn a weighted linear combination of its three layers.

4.2 Self-attention

The model proposed by Mishra et al. (2018b) uses the last hidden state of the RNN as the feature representation for each tweet; instead, we propose the use of a self-attention mechanism to learn a weighted combination of all intermediate hidden states (Rei and Søgaard, 2019). The weights $\hat{a}_i$ for each hidden state h_i are learned by passing h_i through two dense layers with $\tanh$ activation, and a further 1-dimensional dense layer. The final dense layer has either sigmoid or exponential activation, corresponding to soft or sharp attention respectively. The weights are normalised to sum to 1, yielding final attention values $\widetilde{a_i}$, which are used to obtain the final sentential representation $s = \sum_i \widetilde{a_i} h_i$. The RNN is then trained using categorical cross-entropy on s passed through a final $\tanh$ layer.

4.3 RNN Prediction

This modification includes the post-softmax output of the RNN as an additional input feature to the decision tree.

4.4 node2vec

We make use of node2vec to learn low-dimensional continuous representations of hashtags used in tweets on the basis of whole-tweet contexts. We first represent every token (including all hashtags) and each tweet as nodes in a graph, with edges formed between tweets and the tokens they contain. node2vec first follows a tunable sampling strategy to perform random walks from each node, generating directed acyclic graphs with a maximum out degree of 1 (i.e. a sequence of nodes). It then applies the SkipGram model (Mikolov et al., 2013) to learn a representation of each node based on its neighbours in the sampled sequences. Specifically, given a graph with nodes V, node2vec maximises the log probability: $\sum_{v \in V} log(P(N_s(v)|v))$, where $N_s(v)$ is the set of neighboring nodes for node v generated from a sampling strategy s.

We train these node2vec representations on two data sets: the OLID training data, and our own scrape of Twitter using `rtweet` (Kearney, 2018). We collect this additional data by searching for each of the 24 hashtags which appear at least 10 times in the training set, with at least 1 in 4 occurrences in tweets labelled offensive. Intuitively, these common and frequently offensive hashtags are a more reliable signal of offensiveness than less frequent hashtags. It remains to be seen whether collecting more tweets with all hashtags in OLID would help, but the strict rate limits on the Twitter API meant that we ran out of time to explore this.

We trained 200-dimensional embeddings on a random sample of $10,000$ of the resulting tweets. To represent each tweet we sum the embeddings of each hashtag present, and normalised the re-

[1]https://allennlp.org/elmo

System	F1 (macro)
Vanilla model	0.710
Vanilla model + ELMo	0.742
Vanilla model + ELMo + self attention	0.764
Vanilla model + ELMo + self attention + char. ngrams	0.763
Vanilla model + ELMo + self attention + node2vec	0.764
Vanilla model + ELMo + self attention + char. ngrams + node2vec	**0.767**

Table 2: Ablation test for features, with results reported on our held-out development set for subtask A.

sulting vector to unit length. These vectors, or a 200-dimensional 0-vector for tweets containing no hashtags with trained embeddings, were then concatenated with the RNN features (either from self-attention where it was used, or the last hidden state if not) prior to being input into the GBDT.

5 Results

In this section, we present a sample of results obtained during model selection, and results on each of the official subtask test sets. Model selection is carried out by evaluating each model on a consistent 90% training and 10% validation split of the provided training data. Before carrying out model selection, we ran an unmodified version of Mishra et al. (2018b)'s model on subtask A and found that 300-dimensional FastText embeddings trained on Common Crawl gave the best performance[2].

We submitted three models for each of the three subtasks. We submit models that differ in two ways. The first is the amount of data they are trained on. Models labelled ALL-DATA are trained on all of the provided data, while models tagged TRAIN-SPLIT are trained on just the 90% training split, but have a known performance via their results on the development set. It is beneficial to know this as there is a large amount of variance in model results due to stochasticity in the training process. The second way in which the models differ is designed to handle this variance by ensembling three models via majority vote. Such submissions are labelled with ENSEMBLE, while those only using a single model are labelled BEST.

In all three subtasks we find that the best performing system is that which ensembles three identical models trained on the entire training set.

5.1 Subtask A

Subtask A concerns classifying a tweet as OFF (offensive) or NOT (see Section 3). We experiment with adaptations of the model from Mishra et al. (2018b) to perform a 5-WAY classification between all categories, and select the most effective feature combination for each subtask. We experiment with features mentioned in Section 4: ELMo, self-attention, character n-grams, and node2vec. Results from ablation studies are presented in Table 2.

We found that the best performing model used features extracted from a RNN that used ELMo embeddings in addition to FastText and compositional character-based word embeddings, with sharp self-attention over a GRU with 256 hidden units trained using a batch size of 64. These features were used in a GBDT alongside the 10,000 most frequently occurring character n-grams, and node2vec representations of the tweets.

Table 3 shows our results for subtask A on the test data. All three submissions use the model architecture and hyperparameters described above.

System	F1	Accuracy
All NOT baseline	0.419	0.721
All OFF baseline	0.218	0.279
TRAIN-SPLIT-BEST	0.776	0.835
ALL-DATA-ENSEMBLE	**0.788**	**0.847**
ALL-DATA-BEST	0.769	0.835

Table 3: Accuracy and macro F1 results on the official subtask A test set. All three models have the same hyperparameters.

5.2 Subtask B

Subtask B involves a binary classification of whether a tweet is untargeted (UNT) or targeted (TIN). Following Subtask A, we maintain a finer grained classifier using a 4-WAY classification (TIN, IND, GRP, OTH), where we classify a tweet as targeted if the probability for TIN is less than the sum of probabilities for the 3 other labels.

We re-ran feature selection experiments to op-

[2]https://fasttext.cc/docs/en/english-vectors.html

System	F1	Accuracy
All TIN baseline	0.470	0.888
All UNT baseline	0.101	0.113
TRAIN-SPLIT-BEST	0.577	0.717
ALL-DATA-ENSEMBLE	**0.663**	**0.904**
TRAIN-SPLIT-ENSEMBLE	0.657	0.900

Table 4: Accuracy and macro F1 results on the official subtask B test set.

timise for this task. Development experiments showed that the use of character n-grams does not improve performance on this subtask, LSTM performs better than a GRU, and that reducing the RNN dimension to 64 and training batch size to 32 is beneficial. These smaller hyperparameter values are likely more suitable due to the smaller amount of available training data. We found that training node2vec using the provided training data, rather than the scraped dataset, gave better representations, with F1 scores on our held-out development set of 0.635 for OLID data and 0.618 for the extra tweets we obtained from Twitter's API (section 4.4).

Results on the test set are presented in Table 4, where we once again find that the ensemble of classifiers trained on all of the data performs best.

5.3 Subtask C

Subtask C involves classifying the target of an offensive tweet as either an individual, group, or other. As this is the last subtask, only classification between these three labels is possible: there are no finer-grained labels that can be trained on. We find that the best performing model is the same as that in subtask B, except that a GRU is used and the softmax from the RNN is included in the GBDT. Results on the test data are presented in Table 5.

System	F1	Accuracy
All GRP baseline	0.179	0.366
All IND baseline	0.213	0.470
All OTH baseline	0.094	0.164
ALL-DATA-ENSEMBLE	**0.554**	0.704
TRAIN-SPLIT-ENSEMBLE	0.544	**0.709**
TRAIN-SPLIT-BEST	0.534	0.695

Table 5: Accuracy and macro F1 results on the official subtask C test set.

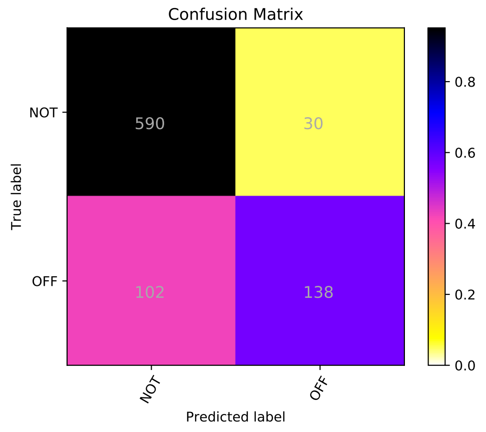


Figure 1: Subtask A, ALL-DATA-ENSEMBLE model.

6 Discussion

In all subtasks, our best performing submission was an ensemble of three identical models, independently trained on all of the training data. Ensembling helps to account for the high variance observed during model training, which occurred despite fixing random seeds.

Across all subtasks we find the inclusion of node2vec features to be helpful. These features offer contextualised representations of hashtags in terms of the tokens they appear with across the corpus, suggesting that features that share information between tweets are useful in addition to those derived from each individually.

We observe that performance drops from subtask A to C. This could be due to the decreasing amounts of training data, from 13,240 instances in Subtask A, to 4,400 in subtask B and 3,876 in subtask C. Very small amounts of data are available for two classes in particular – untargeted offence (UNT) with only 524 training instances, and offence targeted at those other than individuals and groups (OTH) with 395.

As seen in Figures 1 and 2, our model achieves high recall for the NOT class (0.952) in subtask A and for TIN (0.986) in subtask B, but low recall for the other classes OFF (0.575) and UNT (0.259). Figure 3 shows that in subtask C we perform worst on the OTH label, with a low recall of 0.086. In all cases, the model shows weakest performance on the classes for which we have least training data. Therefore, we expect that model performance would improve given more training instances of the minority classes.

Furthermore, in subtask C, the definition of the 'other' class is less clear-cut than the other two cat-

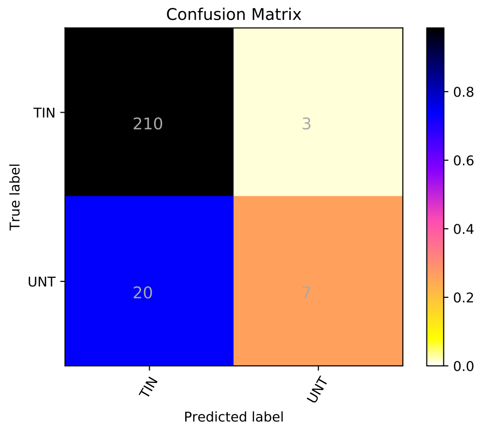


Figure 2: Subtask B, ALL-DATA-ENSEMBLE model.

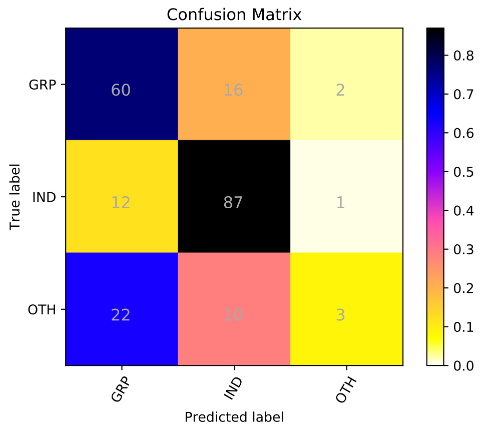


Figure 3: Subtask C, ALL-DATA-ENSEMBLE model.

egories GRP and IND, including abstract concepts such as events or issues, and serving as a catch-all for targeted insults against anything other than specific people or groups of people with a common characteristic. A manual inspection of the data suggests that a large amount of the OTH data includes politically-motivated insults, though similar language also appears in the two other categories, which may make classification harder.

7 Conclusion

The CAMsterdam team attempted the OffensEval tasks taking inspiration from the approach of Mishra and colleagues (2018b), feeding the pre-softmax activation layer of an RNN into a GBDT to classify tweets into one of the applicable fine-grained classes for each subtask. The probabilities of the fine-grained classes were summed to obtain a probability for the desired class: for instance, in subtask A, we summed the probabilities of UNT, IND, GRP and OTH, and compared this sum with the probability of NOT to classify a tweet as offensive or not.

We extended the work of Mishra et al. by using ELMo embeddings as additional input to the RNN, and incorporating a self-attention mechanism following Rei and Søgaard (2019). We also used node2vec to train graph-based representations of hashtags, using both tweets from the OLID training set and new data obtained from the Twitter API featuring hashtags frequently found in the offensive subset. We focus on hashtags on the intuition that they are employed by users to reach those interested in similar topics, and are thus indicative of tweet content. Their use encodes this useful information directly, which we show to be useful for classification. We take into account the fact that hashtags are used in many positions in a tweet by constructing the graph based on co-occurrence across the whole tweet, rather than only within a small window as other embedding methods do.

During development, we found that our best performing models were those formed from an ensemble of three models trained in an identical fashion, thereby smoothing random variation in the training process. The results of the test phase show that our model performed in line with expectations set during development, with F1-scores which decrease from subtask A to C, and lowest precision and recall on the minority classes.

In the future, we will seek to address the imbalance in the training data, inspect the tweets further to analyse the linguistic differences between targeted and untargeted insults, group- and individual-targeted insults and so on. Further architectural changes include collecting more instances of hashtags frequently found in offensive tweets as extra unsupervised data, and we can seek to include author embeddings, a technique found to greatly improve the performance of Mishra et al's system (Mishra et al., 2018a). Finally, we would aim to evaluate our model on other offensive text classification datasets, to discover how well the design generalizes beyond OLID.

Acknowledgements

The 2nd author is supported by the EPSRC, U.K. The 4th, 5th, 6th and 8th authors are members of the ALTA Institute, supported by Cambridge Assessment, University of Cambridge. We thank the NVIDIA Corporation for the donation of the Titan GPU used in this research.

References

Peter Bourgonje, Julian Moreno-Schneider, Ankit Srivastava, and Georg Rehm. 2018. Automatic classification of abusive language and personal attacks in various forms of online communication. In *Language Technologies for the Challenges of the Digital Age*. Springer International Publishing.

Andrew Caines, Sergio Pastrana, Alice Hutchings, and Paula Buttery. 2018. Aggressive language in an online hacking forum. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*. Association for Computational Linguistics.

Justin Cheng, Michael Bernstein, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. 2017. Anyone can become a troll: Causes of trolling behavior in online discussions. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*.

Justin Cheng, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. 2015. Antisocial behavior in online discussion communities. In *The 9th International AAAI Conference on Web and Social Media (ICWSM)*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN EncoderDecoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Karthik Dinakar, Roi Reichart, and Henry Lieberman. 2011. Modeling the detection of textual cyberbullying. In *Fifth International AAAI Conference on Weblogs and Social Media*.

Darja Fišer, Tomaž Erjavec, and Nikola Ljubešić. 2017. Legal framework, dataset and annotation schema for socially unacceptable online discourse practices in Slovene. In *Proceedings of the First Workshop on Abusive Language Online*.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using convolutional neural networks to classify hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3146–3154. Curran Associates, Inc.

Michael W. Kearney. 2018. *rtweet: Collecting Twitter Data*. R package version 0.6.7.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: a Method for Stochastic Optimization. In *International Conference on Learning Representations*, pages 1–13.

Irene Kwok and Yuzhou Wang. 2013. Locate the hate: Detecting tweets against blacks. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Tai Ching Li, Joobin Gharibshah, Evangelos E. Papalexakis, and Michalis Faloutsos. 2017. TrollSpot: Detecting misbehavior in commenting platforms. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*. ArXiv: 1301.3781.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Pushkar Mishra, Marco Del Tredici, Helen Yannakoudakis, and Ekaterina Shutova. 2018a. Author profiling for abuse detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Pushkar Mishra, Helen Yannakoudakis, and Ekaterina Shutova. 2018b. Neural character-based composition models for abuse detection. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, Brussels, Belgium. Association for Computational Linguistics.

Hamdy Mubarak, Kareem Darwish, and Walid Magdy. 2017. Abusive language detection on Arabic social media. In *Proceedings of the First Workshop on Abusive Language Online*.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Marek Rei and Anders Søgaard. 2019. Jointly Learning to Label Sentences and Tokens. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI 2019)*, Honolulu, USA.

Manoel Horta Ribeiro, Pedro H. Calais, Yuri A. Santos, and Wagner Meira J Virgílio A. F. Almeid and. 2018. “Like sheep among wolves”: Characterizing hateful users on Twitter. In *Proceedings of WSDM workshop on Misinformation and Misbehavior Mining on the Web (MIS2)*.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Sara Owsley Sood, Elizabeth F. Churchill, and Judd Antin. 2012. Automatic identification of personal insults on social news sites. *Journal of the American Society for Information Science and Technology*, 63:270–285.

Huei-Po Su, Chen-Jie Huang, Hao-Tsung Chang, and Chuan-Jie Lin. 2017. Rephrasing Profanity in Chinese Text. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the World Wide Web. In *Proceedings of the Second Workshop on Language in Social Media*.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex Machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*.

Jun-Ming Xu, Benjamin Burchfiel, Xiaojin Zhu, and Amy Bellmore. 2013. An examination of regret in bullying tweets. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Dawei Yin, Zhenzhen Xue, Liangjie Hong, Brian D. Davison, April Kontostathis, and Lynne Edwards. 2009. Detection of harassment on Web 2.0. In *Proceedings of the Content Analysis in the WEB 2.0 (CAW2.0) Workshop at WWW2009*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

CN-HIT-MI.T at SemEval-2019 Task 6: Offensive Language Identification Based on BiLSTM with Double Attention

Yaojie zhang, Bing Xu, Tiejun Zhao
Laboratory of Machine Intelligence and Translation, Harbin Institute of Technology
School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
yjzhang@hit-mtlab.net, hitxb@hit.edu.cn, tjzhao@hit.edu.cn

Abstract

Offensive language has become pervasive in social media. In Offensive Language Identification tasks, it may be difficult to predict accurately only according to the surface words. So we try to dig deeper semantic information of text. This paper presents use an attention-based two layers bidirectional long-short memory neural network (BiLSTM) for semantic feature extraction. Additionally, a residual connection mechanism is used to synthesize two different deep features, and an emoji attention mechanism is used to extract semantic information of emojis in text. We participated in three sub-tasks of SemEval 2019 Task 6 as CN-HIT-MI.T team. Our macro-averaged F1-score in sub-task A is 0.768, ranking 28/103. We got 0.638 in sub-task B, ranking 30/75. In sub-task C, we got 0.549, ranking 22/65. We also tried some other methods of not submitting results.

1 Introduction

Recognition of Offensive information has research and application value in many aspects. With the popularity of social media, people's comments on social media has become an important part of public opinion. Although freedom of speech is advocated, there are still some unacceptable words. The study of offensive language has only recently arisen. With the deepening of the research, we need to consider the different sub-tasks of its decomposition.

In OffensEval tasks (Zampieri et al., 2019b), offensive content was divided into three sub-tasks taking the type and target of offenses into account. Sub-task A is offensive language identification. We should identify a short text sentence as offensive or non-offensive. Sub-task B is automatic categorization of offense types. We need to classify a sentence as having an attack target or not, if this is an offensive sentence. Sub-task C is offense target identification. Its purpose is to identify the target of an attack sentence with an attack target. The target is individual, group or other.

User's comments on Twitter are usually cluttered. In order to clean up the data, a series of preprocessing for the data is necessary. Then, pre-trained word vectors are helpful to extract semantic features from deep learning model. For classification model, bidirectional long-short memory neural network can catch the contextual information in text, in order to get offensive semantics from text. A residual connection cascade the first layer's output and the second layer's output can get features of text at different levels. Attention mechanism is used for the final output. Besides, referring to ZHANG et al. (2018), emojis in a sentence have a significant impact on sentiment. We assume them may affect the offensive semantics of text too, and double attention mechanism can deal with this semantic relationship.

The rest of this paper is organized as follows. Section 2 introduces some research advances in Aggression Identification and Hate Speech. Section 3 describes the data we use and the detailed introduction of our system. Section 4 shows the performance of our system and comparison with other models. Section 5 describes some of our summaries and future work directions.

2 Related Work

Due to the universality of offensive language in social media, in order to cope with offensive language and prevent abuse in social media, research on related aspects has gradually emerged in recent years.

There have been several seminars on offensive language research, such as TRAC[1] which shared task on Aggression Identification summarized in Kumar et al. (2018), need to distinguish open, secret and non-aggressive texts. And ALW[2] which is work for Abusive Language. Fišer et al. (2017) did some work on the legal framework, dataset and annotation schema of socially unacceptable discourse practices on social networking platforms in Slovenia. Gambäck and Sikdar (2017) introduced a deep learning based Twitter Hate Speech text include racism, sexism, both and non-hate-speech classification system. Waseem et al. (2017) put forward a series of subtasks of hate speech, cyberbullying, and online abuse. Su et al. (2017) described a system for detecting and modifying Chinese dirty sentences. And GermEval is also shared related tasks (Wiegand et al., 2018) which initiate and foster research on the identification of offensive content in German language microposts. Additionally, the main work of Malmasi and Zampieri (2017) and Malmasi and Zampieri (2018) is to approach the problem of distinguishing general profanity from hate speech. Zhang et al. (2018) tried to use a Convolution-GRU for detecting hate speech on Twitter.

3 Methodology and Data

3.1 Method

Our method is composed of 6 stages: Preprocessing, Embedding, Bidirectional LSTM, Attention, Double attention and Softmax. The whole architecture of the single model is shown in Figure 1.

3.1.1 Preprocessing

We have done some processing on the original training data and test data. The main purpose is to make the data cleaner, reduce the number of unknown words in the dictionary, and do some processing of error words. The first step is to convert all words into lowercase. Our preliminary view is that uppercase or lowercase has no direct impact on offensive language recognition tasks. The second step is to deal with punctuation symbols. We use space as separator to divide sentences into words. But in many cases in data sets, punctuation symbols and words, as well as punctuation symbols and punctuation symbols are closely linked without space. In this way, the system will not be able to recognize them. Just like ”sloth.” or ”!!!”, and we turned them into ”sloth .” and ”! ! !”. The third step is abbreviation processing. We converted ”don't”, ”you're” and so on into ”do n't” and ”you 're”, because there is no ”don't” or ”you're” in the dictionary. We haven't expanded the abbreviations like ”do not” or ”you are”, because the results may not be unique just like ”I'd” can represent ”I would” or ”I had”. Forth, we also need to separate the emojis refer to the dictionary of emojis. (We will introduce the word dictionary and emoji dictionary in detail in embedding section). In addition, we regard all numbers as one word.

After completing the above preprocessing, if a word is still detected as an unknown word, we use ekphrasis[3] tool (Baziotis et al., 2017) for further processing. The first step is word segmentation. We separate some words together may be a customary expression by spaces. For example, ”Googlearecorrupt” is turned into ”Google are corrupt”. Second, if the word still does not exist in the dictionary, we do an error correction operation. After all operations, words that do not exist in the dictionary are marked as unknown words.

Step-by-step processing instead of direct batch processing without intermediate detection improves reliability and avoids modifying correct expressions to errors.

3.1.2 Embedding Layer

We used a word embedding layer to represent words as vectors. The 200 dimension word vectors[4] is pre-trained by GloVe based on a large corpus of Twitter provided by Jeffrey et al. (2014). It includes 1,193,514 words and their vector representations.

A sentence sequence $S(w_1, w_2, ..., w_l)$ is represented as $C(c_1, c_2, ..., c_l)$, where $w_i (i \in [1, l])$ represents a word, c_i represents the vector corresponding to the word. C is the matrix representation of the sequence S.

We also use emoji vectors provided by Eisner et al. (2016) to represent the emoji that appears in a sequence. It includes 1,661 emojis used in

[1] https://sites.google.com/view/trac1/home

[2] https://sites.google.com/site/abusivelanguageworkshop2017/

[3] https://www.github.com/cbaziotis/ekphrasis

[4] https://nlp.stanford.edu/projects/glove/

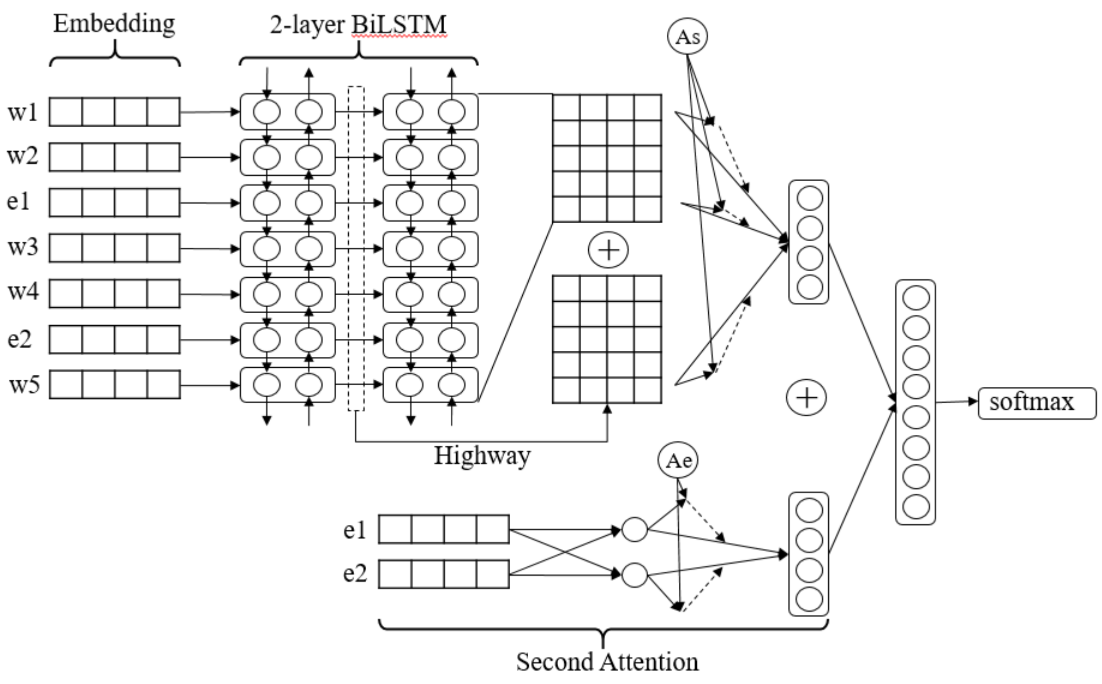


Figure 1: The whole architecture of 2-layer BiLSTM with Double Attention Based Offensive Language Identification. Where w is word, and e is emoji

Twitter and their 300 dimension vector representations. We have made a PCA dimension reduction on emoji vectors, making 300 dimensions into 200 dimensions to suit word vectors. In addition, all bottom feature representation vector are normalized.

3.1.3 Bidirectional LSTM Layer

The structure of one cell in LSTM is shown in Figure 2. Three gated mechanisms of LSTM allows LSTM memory unit to store and access information for a long time. i_t express input gate, f_t express forget gate and o_t express output gate. c_t express memory cell, s_t express memory state and h_t express hidden state. Where t denotes at time t. The forward pass of LSTM is shown in (1)-(6).

$$i_t = f(W_i x_t + U_i h_{t-1} + V_i c_{t-1}) \tag{1}$$

$$f_t = f(W_f x_t + U_f h_{t-1} + V_f c_{t-1}) \tag{2}$$

$$o_t = f(W_o x_t + U_o h_{t-1} + V_o c_t) \tag{3}$$

$$c_t = g(W_c x_t + U_C h_{t-1}) \tag{4}$$

$$s_t = f_t \odot c_{t-1} + i_t \odot c_t \tag{5}$$

$$h_t = o_t \odot g(c_t) \tag{6}$$

Where x_t is the input at time t, $f(.)$ is the sigmoid function, $g(.)$ is the hyperbolic function. W, U and V are trainable weight parameters. In order to extract the semantic relationship features of

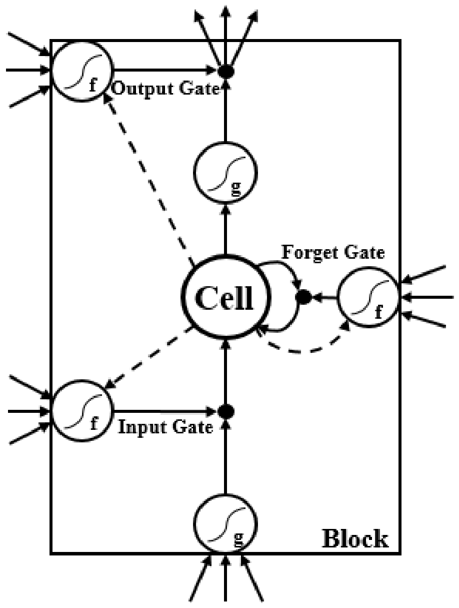


Figure 2: The structure of one cell in LSTM

sentences before and after, we use Bidirectional LSTM to process a sentence. Forward and backward LSTM obtains hidden states $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$:

$$\overrightarrow{h_t} = \overrightarrow{LSTM}\ (w_t, \overrightarrow{h_{t-1}}) \tag{7}$$

$$\overleftarrow{h_t} = \overleftarrow{LSTM}\ (w_t, \overleftarrow{h_{t-1}}) \tag{8}$$

Cascade the results of bidirectional hidden state as the result of BiLSTM:

$$H_t = \overrightarrow{h_t} \oplus \overleftarrow{h_t} \tag{9}$$

We stack two layers of BiLSTMs. The output of the first layer BiLSTM is used as the input of the second layer. Meanwhile, we consider that the first layer can collect low-level semantic information such as lexical or grammatical information, while the second layer can collect high-level semantic information such as sentiment or offensiveness information. So we added a residual connection between the two layers:

$$H_t^{final} = H_t^{layer_1} \oplus H_t^{layer_2} \tag{10}$$

3.1.4 Attention Layer

We input the representation of hidden state obtained by Bidirectional LSTM layer into attention layer to get sentence coding:

$$s_i = \sum_t \alpha_{it} H_{it}^{final} \tag{11}$$

Where a is the weight value:

$$\alpha_{it} = \frac{exp(u_{it}^{\mathsf{T}} u_w)}{\sum_t exp(u_{it}^{\mathsf{T}} u_w)} \tag{12}$$

$$u_{it} = tanh(W_w H_{it}^{final} + b_w) \tag{13}$$

3.1.5 Double Attention Mechanism

We use another attention mechanism similar to that of encoding sentences to encode emojis in sentences, referring to (ZHANG et al., 2018). If there are emojis in a sentence, we get the corresponding vector representation from the emojis dictionary mentioned in Section 3.2.2. Then the coding is obtained through the attention mechanism:

$$s_i^e = \sum_t \alpha_{it}^e E_i \tag{14}$$

Where E is the vector representation of emojis in a sentence.

3.1.6 Softmax Layer

Finally, we concatenate sentence coding and emojis coding through the full connection layer. We use the softmax classifier to construct scoring vectors for each category and convert them into probabilities:

$$r = s \oplus s_e \tag{15}$$

$$\hat{y} = \frac{exp(Wr + b)}{\sum_{i \in [1,l]} exp(W_i r + b_i)} \tag{16}$$

Where W and b is the layer's weights and biases. We use cross-entropy loss function with L2 regularization term:

$$L(\hat{y}, y) = -\sum_{i=1}^{N} \sum_{j=1}^{C} y_i^j log(\hat{y_i^j}) + \lambda(\sum_{\theta \in \Theta} \theta^2) \tag{17}$$

3.2 Data

We only use the training dataset which contains 13,240 tweets provided by SemEval 2019 Task 6 (Zampieri et al., 2019a). Table 1 shows three different levels of tasks and their corresponding amount of data.

A	B	C	Train	Test	Total
OFF	TIN	IND	2,407	100	2,507
OFF	TIN	OTH	395	35	430
OFF	TIN	GRP	1,074	78	1,152
OFF	UNT	-	524	27	551
NOT	-	-	8,840	620	9,460
All			13,240	860	14,100

Table 1: Distribution of label combinations in the data provided by OLID

We have 13,240 tweets to train subtask A. Of these, 4,400 are offensive and 8,840 are non-offensive. There are 4,400 tweets for subtask B. Of these, 3,876 are targeted insult and threats and 524 are untargeted. In 3,876 targeted tweets, 2,407 of them are individual, 1,074 of them are group, and 395 of them are other.

4 Results

In this part, some experimental settings are briefly described. Additionally, we list the experimental results we submitted and compared them with baseline. We use 5-fold cross validation to get 5 same models. Using the probability predicted by these 5 models to do a soft-vote to get the final probability distribution. The highest probability is the predictive class of our system. What's more, we compare performance on offensive identification detection of different models by 5-fold cross validation.

The number of hidden units in 2-layer of BiLSTM is 150 each layer for sub-task A, where it is 100 for sub-task B and 80 for sub-task C. The size of randomly initialized attention query vector is 256 dimensions. We chose the adam optimizer, and the learning rate starts at $7e-4$, decreases to 0.9 times per 20 epoch, always greater than $1e-4$. The λ of L2 regularization is $1e-5$

Tables 2, 3 and 4 show the performance of our system on test dataset in subtasks A, B and C, respectively. The first 2 or 3 rows of the table show the baseline of the officially provided subtasks.

System	F1 (macro)	Accuracy
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
Submitted System	**0.7684**	**0.8244**

Table 2: The results of Sub-task A we submitted. The system is a 2-layer BiLSTM with Double Attention which is described in Section 3.2

System	F1 (macro)	Accuracy
All TIN baseline	0.4702	**0.8875**
All UNT baseline	0.1011	0.1125
Submitted System	**0.6381**	0.8417

Table 3: The results of Sub-task B we submitted. The system is a 2-layer BiLSTM with Double Attention which is described in Section 3.2. It has different hyperparameter settings from Sub-task A

System	F1 (macro)	Accuracy
All GRP baseline	0.1787	0.3662
All IND baseline	0.2130	0.4695
All OTH baseline	0.0941	0.1643
Submitted System	**0.5488**	**0.6338**

Table 4: The results of Sub-task C we submitted. The system is a 2-layer BiLSTM with Double Attention which is described in Section 3.2. It has different hyperparameter settings from Sub-task A and Sub-task B

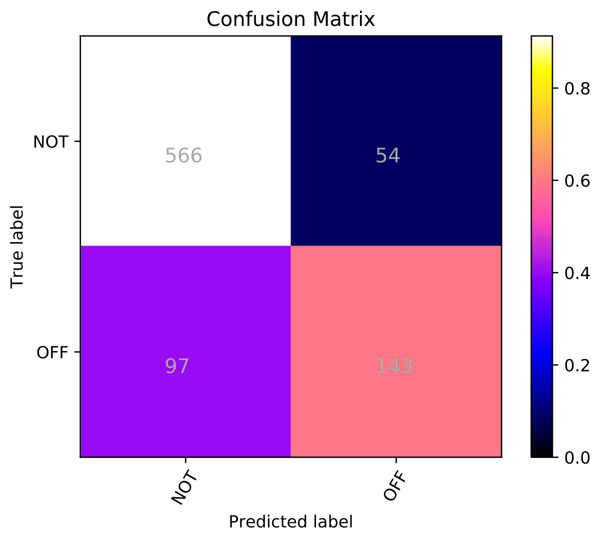


Figure 3: The confusion matrix of the classification results in Sub-task A

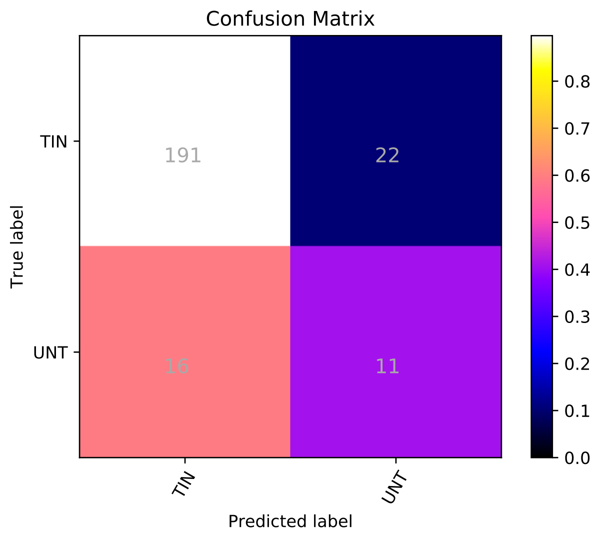


Figure 4: The confusion matrix of the classification results in Sub-task B

Figures 3, 4 and 5 show the confusion matrix of the classification results of our system on the test dataset in subtasks A, B and C, respectively.

We can clearly see from figures that our system performs better in category has more training data. This may be because more data in this category makes the system more inclined to classify these samples correctly in training, not because some categories are easier to identify and others are harder. We used the over-sampling method, but the effect is not obvious. Categories with fewer data quickly reach the state of over-fitting, while those with more data are still under-fitting.

Table 5 shows some comparative experiments on closed validation sets. We use 5-fold cross validation, and each data has the same category ratio.

We can see that if we only use the second layer's features, the performance is worse than using the first layer's. When using two layers features at the same time, the effect is the best. In addition, Word Attention can significantly improve system performance, but Emoji Attention only improve system performance a little. This may be because emojis have little impact on text offensive semantics. It is worth mentioning that the system performance is improved obviously after data preprocessing, and the F1 score has increased by 3.12%.

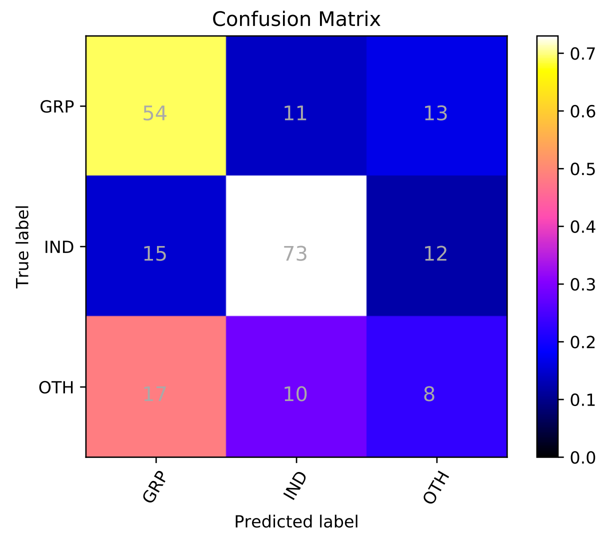


Figure 5: The confusion matrix of the classification results in Sub-task C

Model	A	B	C
1-layer BiLSTM	0.749	0.610	0.527
1-layer BiLSTM with Attention	0.757	0.632	0.544
2-layer BiLSTM with Attention	0.752	0.626	0.513
2-layer BiLSTM (residual connected) with Attention	0.764	0.643	0.549
2-layer BiLSTM (residual connected) with Double Attention (Emojis)	**0.766**	**0.643**	**0.550**

Table 5: Some comparative experiments on closed validation sets

5 Conclusion

This paper introduces a deep learning method Attention-based residual connected BiLSTM with Emojis Attention for SemEval 2019 Task 6: Identifying and Categorizing Offensive Language in Social Media. Our system didn't get the leading score in the competition. Maybe there are the following reasons. Except for the corpus used for pre-training word vectors, we do not use other data for training. Some machine learning models may achieve better results with fewer data. Additionally, we think that hyperparameter adjustment in our system is not perfect. Most importantly, we do not associate the characteristics of offensive language recognition tasks with our model.

The next step of this paper is to associate tasks with models. We will try to constructing a dictionary of offensive words for tasks. We will also try other ways of using external dictionaries except double attention mechanism such as Position Embedding. In addition, we will try some language models that are currently leading the NLP task such as BERT proposed by Devlin et al. (2018).

References

C Baziotis, N Pelekis, and C Doulkeridis. 2017. DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv e-prints*, page arXiv:1810.04805.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning Emoji Representations from their Description. In Proceedings of the 4th International Workshop on Natural Language. In *Proceedings of the 4th International Workshop on Natural Language Processing for Social Media at EMNLP 2016 (SocialNLP at EMNLP 2016)*.

Darja Fišer, Tomaž Erjavec, and Nikola Ljubešić. 2017. Legal Framework, Dataset and Annotation Schema for Socially Unacceptable On-line Discourse Practices in Slovene. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Pennington Jeffrey, Socher Richard, and D. Manning Christopher. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 467–472.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Huei-Po Su, Chen-Jie Huang, Hao-Tsung Chang, and Chuan-Jie Lin. 2017. Rephrasing Profanity in Chinese Text. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Yangsen ZHANG, Jia ZHENG, Gaijuan HUANG, and et al. 2018. Microblog sentiment analysis method based on a double attention model. In *Proceedings of Tsinghua University(Science and Technology)*, volume 58, pages 122–130.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

ConvAI at SemEval-2019 Task 6: Offensive Language Identification and Categorization with Perspective and BERT

John Pavlopoulos
Ion Androutsopoulos
Department of Informatics
Athens University of Economics
and Business, Greece
`annis,ion@aueb.gr`

Nithum Thain
Lucas Dixon
Jigsaw
`nthain,ldixon@google.com`

Abstract

This paper presents the application of two strong baseline systems for toxicity detection and evaluates their performance in identifying and categorizing offensive language in social media. **Perspective** is an API, that serves multiple machine learning models for the improvement of conversations online, as well as a toxicity detection system, trained on a wide variety of comments from platforms across the Internet. **BERT** is a recently popular language representation model, fine tuned per task and achieving state of the art performance in multiple NLP tasks. **Perspective** performed better than **BERT** in detecting toxicity, but **BERT** was much better in categorizing the offensive type. Both baselines were ranked surprisingly high in the SEMEVAL-2019 OFFENSEVAL competition, **Perspective** in detecting an offensive post (12th) and **BERT** in categorizing it (11th). The main contribution of this paper is the assessment of two strong baselines for the identification (**Perspective**) and the categorization (**BERT**) of offensive language with little or no additional training data.

1 Introduction

Offensive language detection refers to computational approaches for detecting abusive language, such as threats, insults, calumniation, discrimination, swearing (Pavlopoulos et al., 2017b), which could be targeted (at an individual or group) or not (Waseem et al., 2017). These computational approaches are often used by moderators who face an increasing volume of abusive content and would like assistance in managing it efficiently.[1]

Although offensive language detection is not a new task (Dinakar et al., 2011; Dadvar et al., 2013; Kwok and Wang, 2013; Burnap and Williams, 2015; Tulkens et al., 2016), the creation of large corpora (Wulczyn et al., 2017), along with recent advances in pre-training text representations (Devlin et al., 2018) allow for much more efficient approaches. Furthermore, while new competitions and corpora are being introduced (Zampieri et al., 2019a),[2] there is a need for strong baselines to assess the performance of more complex systems. This paper assesses two systems for the detection and categorization of offensive language, which require few or no task-specific annotated training instances.

The first baseline is a Convolutional Neural Network (CNN) for toxicity detection, trained on millions of user comments from different online publishers, which is made publicly available through the **Perspective** API.[3] This model requires no extra training or fine tuning and can be directly applied to score unseen posts. The second strong baseline is the recently popular Bidirectional Encoder Representations from Transformers (**BERT**), a pre-trained model that has been reported to achieve state of the art performance in multiple NLP tasks with limited fine-tuning on task-specific training data (Devlin et al., 2018).

Section 2 below summarizes related work and Section 3 discusses the SEMEVAL-2019 OFFENSEVAL dataset we used. In Section 4 we describe the two proposed baselines and we report experimental results in Section 5. Section 6 concludes our work and suggests future directions.

2 Related Work

Various forms of offensive language detection have recently attracted a lot of attention (Nobata et al., 2016; Pavlopoulos et al., 2017b; Park and Fung, 2017; Wulczyn et al., 2017). Apart from the growing volume of popular press concerning

[1]See, for example, `https://goo.gl/VQNDNX`.

[2]See also `https://goo.gl/v7kA1K`.

[3]`https://www.perspectiveapi.com/`

toxicity online, the increased interest in research into offensive language is partly due to the recent Workshops on Abusive Language Online,[4] as well as other fora, such as GermEval for German texts,[5] or TA-COS[6] and TRAC (Kumar et al., 2018),[7]. The literature contains many terms for different kinds of offensive language: toxic, abusive, hateful, attacking, etc. Largely, these are defined by different survey methods. In (Waseem et al., 2017), abusive language is divided into explicit vs. implicit, and directed vs. generalized. However, other researchers have created different taxonomies based on sub-kinds of toxic language (Table 2).

Although some previous research has considered several types of abuse and their relations (Malmasi and Zampieri, 2018), detecting varieties of hate has attracted more attention (Djuric et al., 2015; Malmasi and Zampieri, 2017; ElSherief et al., 2018; Gambäck and Sikdar, 2017; Zhang et al., 2018). The first publicly available dataset for hate speech detection was that of Waseem and Hovy (2016). It contained 1607 English tweets annotated for sexism and racism. A larger dataset was published by Davidson et al. (2017), containing approx. 25K tweets collected by using a hate lexicon. Despite the popularity of hate speech detection in literature, no larger publicly available hate speech datasets seem to exist. For recent overviews of hate speech detection, consult Schmidt and Wiegand (2017) and Fortuna and Nunes (2018).

Research into the various kinds of offensive language detection is mainly focused on English, but some work in other languages also exists. Work on a large dataset of Greek moderated news portal comments is presented by Pavlopoulos et al. (2017a). A dataset of obscene and offensive user comments and words in Arabic social media was presented by Mubarak et al. (2017). Previous work includes a system to detect and rephrase profanity in Chinese (Su et al., 2017), and an annotation schema for unacceptable social media content in Slovene (Fišer et al., 2017).

[4]https://goo.gl/9HmSzc
[5]https://goo.gl/uZEerk
[6]http://ta-cos.org/
[7]https://goo.gl/DTZquU

3 Data

The SemEval-2019 OffensEval dataset that is available to participants contains 13240 tweets; the counts of the labels are shown in Table 1. The OffensEval task consists of three subtasks, described in detail by Zampieri et al. (2019b). Subtask A aims at the detection of offensive language (OFF or NOT in Table 3). Subtask B aims at categorizing offensive language as targeting a specific entity (TIN) or not (UNT). Subtask C aims to identify whether the target of an offensive post is an individual (IND), a group (GRP), or unknown (OTH). Table 1 also shows the size of the vocabulary per class (label), which, unsurprisingly, is proportional to the class size. It is worth noting that offensive tweets targeting a group are the lengthier texts, with 28 tokens on average (see Table 1, column C, GRP column).

4 Baselines

We now describe the two baselines (**Perspective**, **Bert**) that we implemented and evaluated.

4.1 Perspective

We employed the **Perspective** API, which was created by Jigsaw and Google's Counter Abuse Technology team in Conversation-AI,[8] to facilitate better conversations online and protect voices in conversations (Hosseini et al., 2017). Although open-source code is available,[9] we chose to use pre-trained models, accessible through the API. For offensive language detection in Subtask A, we used the *Toxicity* model, which is a CNN based on GloVe word embeddings,[10] trained over millions of user comments from publishers such as the New York Times and Wikipedia. This is a robust model, which we expect to be somewhat adaptable to different datasets (and their labels for closely related forms of offensive language), such as the offensive tweets of OffensEval. For offensive language categorization in Subtask B, we employed other experimental models, also available via the **Perspective** API, which detect various abuse types including those of Table 2.

4.2 BERT

Bert (Devlin et al., 2018) is a deep bidirectional network built using Transformers (Vaswani et al.,

[8]https://conversationai.github.io/
[9]https://goo.gl/yN196H
[10]https://goo.gl/rHYMqt

Subtask	A		B		C		
Label	NOT	OFF	UNT	TIN	IND	GRP	OTH
Number of Tweets	8840	4400	524	3876	2407	1074	395
Class specific vocabulary size	29.2K	18.6K	3.5K	17.3K	11.5K	7.8K	3.5K
Average number of tokens / Tweet	22	24	19	24	22	28	25

Table 1: Number of tweets, size of vocabulary, and average number of tokens per tweet, for each label (class). In Subtask A, the labels are 'not offensive' (NOT) or 'offensive' (OFF). In Subtask B, the labels are 'not targeted threat' (UNT) and 'targeted insult or threat' (TIN). In Subtask C, they are 'targeted insult or threat towards an individual' (IND), 'towards a group' (GRP), or 'towards another target' (OTH).

TOXICITY	@user Fuck you, you fat piece of shit
INSULT	Hey @user , you are disgusting.
THREAT	@user Kill the traitors.
PROFANITY	My wrist been fucked up for nearly a month now . This time im really going to the hospital to see what the fuck is wrong with it
IDENTITY ATTACK	Okay everyone always talks aboht the pathetic army and all the soy boy branches and gay shit and what not [...]
ATTACK ON COMMENTER	@user You are all utterly delusional. If you were really pro-life" you would [...]

Table 2: The tweets with the highest **Perspective** score per abusiveness type, on a trial dataset of 320 tweets shared by the competition organizers.

2017). It is pre-trained to detect (a) a masked word from its left and right context, and (b) the next sentence. We used the publicly available **BERT**-BASE version,[11] with 12 Transformer layers, 768 hidden states size, which is pre-trained on a monolingual corpus of 3.3B words. For a particular NLP task, a task-specific layer is added on top of **BERT**. In our case, the extra layer comprises dropout, a linear transformation, and softmax.[12] During the task-specific 'fine-tuning', the extra layer is trained jointly with **BERT** (refining the pre-trained **BERT** model) on task-specific data. Previous research demonstrated that fine-tuning **BERT** leads to state of the art performance in several NLP tasks (Devlin et al., 2018).

System	**F1 (macro)**	**Accuracy**
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
Perspective	**0.7933**	**0.8360**
BERT	0.7705	0.8163
BEST 2019	0.8290	—

Table 3: Results for Subtask A.

System	**F1 (macro)**	**Accuracy**
All TIN baseline	0.4702	**0.8875**
All UNT baseline	0.1011	0.1125
Perspective	0.4785	0.6292
BERT	**0.6817**	0.8708
BEST 2019	0.7550	—

Table 4: Results for Subtask B.

5 Results

5.1 Offensive Language Detection

For Subtask A, we used the *toxicity* score from **Perspective** and returned the offensive label (OFF) when the returned score was above 0.5. No fine tuning was performed for **Perspective**. For **BERT**, we split the dataset to training (10K tweets) and development (3240) subsets, and fine-tuned **BERT** for 3 epochs.[13]

In this subtask, **Perspective** outperformed **BERT** and was ranked 12th out of 103 submissions. The difference from the top-ranked model was 3.5 F1 points. The performance of **Perspective** in this subtask is particularly interesting, considering that the training data for these models were not labeled for offensiveness, but rather for other attributes such as toxicity, threats, and insults.[14] Ignoring **Perspective**, **BERT** was ranked

[11] https://goo.gl/95mqhE

[12] We used default values for all hyper parameters.

[13] We used the uncased system with batch size 32, based on preliminary experiments.

[14] https://goo.gl/Bmiogb

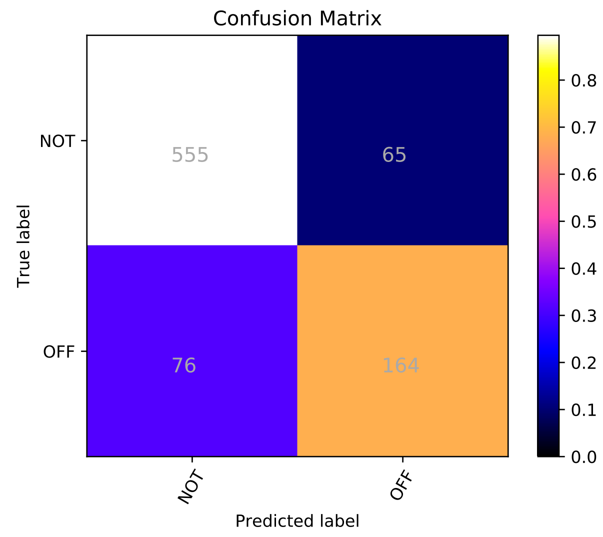


Figure 1: Confusion matrix for **Perspective** in Subtask A (Offensive Language Detection).

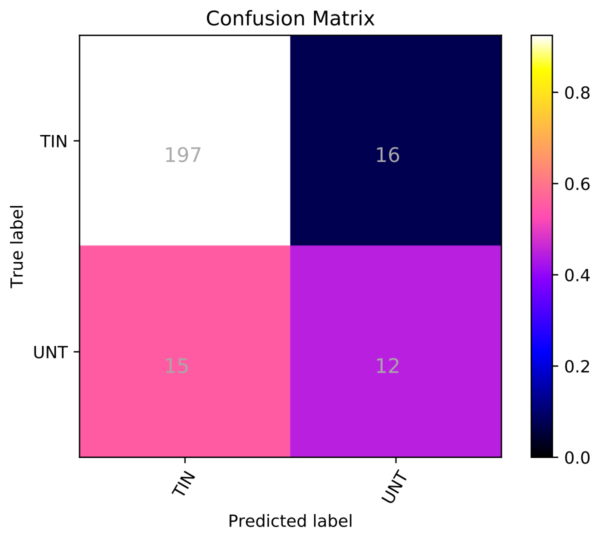


Figure 2: Confusion matrix for **BERT** in Subtask B (Offense Type Detection).

27th. As shown in Table 3, both of our strong baselines outperform the naive majority baselines for this subtask.

The confusion matrix of **Perspective** is shown in Fig. 1. Both recall and precision are high for the NOT label (87.96% and 89.81%), but lower for OFF (68.33% and 71.62%). This is explained by the fact that NOT is two times the size of OFF (Table 1). We also used **Perspective** to score the training data, since no fine-tuning was performed on the training data for **Perspective**. Macro F1 was 78.01% (85.02% for NOT, 71% for OFF) and accuracy was 80.24%, which are lower but close to the respective values on the test data (Table 3).

5.2 Offense Type Detection

For Subtask B, we used the experimental *insult*, *threat* and *attack on commenter* models from Perspective. We averaged *insult* and *attack on commenter* and used this average to compare with the *threat* score. The **Perspective** baseline returned a targeted insult/threat (TIN) when the average was greater, and untargeted (UNT) otherwise. The **BERT** baseline was fine-tuned on the entire dataset that was available to participants, because we considered that dataset too small for a training/development split.[15] **BERT** clearly outperformed the **Perspective** baseline (Table 4) and ranked 11th in this subtask among 73 participants, whereas the best system achieved 7.8 points higher in F1. The confusion matrix of **BERT** for this subtask is shown in Fig. 2. The large class imbalance (TIN tweets are approx. 7 times than UNT, see Table 1) significantly reduces both the recall (44.44%) and precision (42.86%) of **BERT** for the UNT class, compared to TIN (92.49% and 92.92%, respectively).

5.3 Offense Target Detection

For Subtask C, Perspective has no suitable model to respond yet and the **BERT**-based systems submitted were in an experimental phase, due to time constraints.[16] We consider the results we obtained for this subtask as not relevant and leave the development and evaluation of baselines for this subtask as future work.

6 Conclusion

This paper proposed and evaluated two strong baselines, based on the **Perspective** API and **BERT**, for identifying and categorizing offensive language in social media. Both baselines require few (**BERT**) or no additional task-specific training data (**Perspective**) and this is the first work, to our knowledge, to assess their performance in the tasks we considered. The **Perspective**-based baseline was ranked 12th among 103 submissions for the task of classifying a post as offensive or not. The **BERT** baseline was ranked 11th among

[15]We used the cased system with batch size 16, based on preliminary experiments.

[16]**BERT** base and **BERT** large (trained on CPU) were examined for this subtask, but preliminary experiments showed that the majority class was always returned.

73 submissions for the task of recognizing whether an offensive post is targeted or not. Both baselines were ranked surprisingly high in the corresponding tasks, considering that they were given no or few, respectively, additional task-specific training instances. Furthermore, the Perspective baseline, which required no fine tuning outperformed **BERT** by a large margin in the task of detecting offensive language. In future work, we intend to examine stronger, yet easy to apply baselines, and release source to make it easier to use them.

References

P. Burnap and M. L. Williams. 2015. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet*, 7(2):223–242.

M. Dadvar, D. Trieschnigg, R. Ordelman, and F. de Jong. 2013. Improving cyberbullying detection with user context. In *European Conference on Information Retrieval*, pages 693–696.

T. Davidson, D. Warmsley, M. Macy, and I. Weber. 2017. Automated hate speech detection and the problem of offensive language. In *ICWSM*, pages 512–515, Montreal, Canada.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv*.

K. Dinakar, R. Reichart, and H. Lieberman. 2011. Modeling the detection of textual cyberbullying. In *The Social Mobile Web*, pages 11–17.

N. Djuric, J. Zhou, R. Morris, M. Grbovic, V. Radosavljevic, and N. Bhamidipati. 2015. Hate speech detection with comment embeddings. In *ICWWW*, pages 29–30.

M. ElSherief, V. Kulkarni, D. Nguyen, W. Y. Wang, and E. Belding. 2018. Hate lingo: A target-based linguistic analysis of hate speech in social media. *arXiv preprint*.

D. Fišer, T. Erjavec, and N. Ljubešić. 2017. Legal framework, dataset and annotation schema for socially unacceptable on-line discourse practices in slovene. In *1st Workshop on Abusive Language Online*, Vancouver, Canada.

P. Fortuna and S. Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51(4):85.

B. Gambäck and U. K. Sikdar. 2017. Using convolutional neural networks to classify hate-speech. In *1st Workshop on Abusive Language Online*, pages 85–90, Vancouver, Canada.

H. Hosseini, S. Kannan, B. Zhang, and R. Poovendran. 2017. Deceiving google's perspective api built for detecting toxic comments. In *arXiv preprint*.

R. Kumar, A. K. Ojha, S. Malmasi, and M. Zampieri. 2018. Benchmarking aggression identification in social media. In *TRAC*, Santa Fe, USA.

I. Kwok and Y. Wang. 2013. Locate the hate: Detecting tweets against blacks. In *AAAI*, pages 1621–1622, Whasington, USA.

S. Malmasi and M. Zampieri. 2017. Detecting hate speech in social media. In *RANLP*, pages 467–472.

S. Malmasi and M. Zampieri. 2018. Challenges in discriminating profanity from hate speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

H. Mubarak, K. Darwish, and W. Magdy. 2017. Abusive language detection on arabic social media. In *1st Abusive Language Workshop*, pages 52–56, Vancouver, Canada.

C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang. 2016. Abusive language detection in online user content. In *ICWWW*, pages 145–153.

J. H. Park and P. Fung. 2017. One-step and two-step classification for abusive language detection on twitter. In *1st Workshop on Abusive Language Online*, pages 41–45.

J. Pavlopoulos, P. Malakasiotis, and I. Androutsopoulos. 2017a. Deep learning for user comment moderation. In *1st Workshop on Abusive Language Online*, pages 25–35.

J. Pavlopoulos, P. Malakasiotis, and I. Androutsopoulos. 2017b. Deeper attention to abusive user content moderation. In *EMNLP*, pages 1125–1135, Copenghagen, Denmark.

A. Schmidt and M. Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia, Spain.

H.-P. Su, C.-J. Huang, H-T. Chang, and C.-J. Lin. 2017. Rephrasing profanity in chinese text. In *1st Workshop on Abusive Language Online*, Vancouver, Canada.

S. Tulkens, L. Hilte, E. Lodewyckx, B. Verhoeven, and W. Daelemans. 2016. A dictionary-based approach to racism detection in dutch social media. In *TA-COS*, Portoroz, Slovenia.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.

Z. Waseem, T. Davidson, D. Warmsley, and I. Weber. 2017. Understanding abuse: A typology of abusive language detection subtasks. In *1st Workshop on Abusive Langauge Online*, Vancouver, Canada.

Z. Waseem and D. Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *NAACL SRW*, pages 88–93, San Diego, California.

E. Wulczyn, N. Thain, and L. Dixon. 2017. Ex machina: Personal attacks seen at scale. In *ICWWW*, pages 1391–1399.

M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *NAACL*.

M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar. 2019b. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). In *SemEval*.

Z. Zhang, D. Robinson, and J. Tepper. 2018. Detecting hate speech on twitter using a convolution-gru based deep neural network. In *Lecture Notes in Computer Science*. Springer Verlag.

DA-LD-Hildesheim at SemEval-2019 Task 6: Tracking Offensive Content with Deep Learning using Shallow Representation

Sandip Modha
DA-IICT
Gandhinagar India
sjmodha
@gmail.com

Prasenjit Majumder
DA-IICT
Gandhinagar India
p_majumder
@daiict.ac.in

Thomas Mandl
Uni. of Hildesheim
Hildesheim Germany
mandl
@uni-hildesheim.de

Daksh Patel
LDRP-ITR
Gandhinagar India
dakshpatel68
@gmail.com

Abstract

This paper presents the participation of team DA-LD-Hildesheim of Information Retrieval and Language Processing lab at DA-IICT, India in Semeval-19 OffenEval track. The aim of this shared task is to identify offensive content at fined-grained level granularity. The task is divided into three sub-tasks. The system is required to check whether social media posts contain any offensive or profane content or not, targeted or untargeted towards any entity and classifying targeted posts into the individual, group or other categories. Social media posts suffer from data sparsity problem, Therefore, the distributed word representation technique is chosen over the Bag-of-Words for the text representation. Since limited labeled data was available for the training, pre-trained word vectors are used and fine-tuned on this classification task. Various deep learning models based on LSTM, Bidirectional LSTM, CNN, and Stacked CNN are used for the classification. It has been observed that labeled data was highly affected with class imbalance and our technique to handle the class-balance was not effective, in fact performance was degraded in some of the runs. Macro F1 score is used as a primary evaluation metric for the performance. Our System achieves Macro F1 score = 0.7833 in sub-task A, 0.6456 in the sub-task B and 0.5533 in the sub-task C.

1 Introduction

NLP researchers are developing innovative systems based on the input of the text data. The power of predictions has moved from simple sentiment classification task to much more advanced labeling of the content. The task related to hate, aggression, abusive or offensive speech currently attracts research more to algorithms making decisions which can also be ambiguous for humans. Due to the availability of standard datasets, such data collections are created based on social media data and are offered at forum like TRAC [1] (Kumar et al., 2018), GermEval [2], and SemEval OffenEval 2019 (Zampieri et al., 2019a)[3].

The exponential rise in social media user-base backed by the cutting edge mobile data technologies leads to the inorganic growth in the posts related to hate speech or offensive speech. Researchers working in the area of domain-specific sentiment analysis move to the problem of domain specific or open domain hate or offensive speech detection. They are reshaping the hate speech problem into the new notion like abusive, aggressive, or offensive speech. Such categorization of social media posts, help law-enforcement agencies with the surveillance of the social media.

The shared task in SemEval-OffenEval 2019 was introduced as a 3-level classification task (Zampieri et al., 2019b). In the first level, sub-task A, systems are required to classify tweets into two class, namely: Offensive (OFF) and Non-offensive (NOT). In the second level, sub-task B, offensive tweets are further required to be categorized into two labels, namely :targeted (TIN)-post which contain threat/insult to the targeted entity and untargeted (UNT), respectively. In the sub-task C, target of insults and threats are further classified to Individual (IND), Group (GRP), or Other (OTH) classes. Table 1 presents the statistic about the dataset. One can observe that classes in dataset, particularly for the sub-task B and sub-task C, is highly imbalanced.

Our approach for this shared task is based on distributed word representation and deep learning. fastText pre-trained word embedding (Mikolov et al., 2018) is used to initialize embedding layer or first layer of the model and fine tuned for classification task. The rest of the model is still needed

[1]https://sites.google.com/view/trac1/home
[2]http://https://projects.fzai.h-da.de/iggsa/
[3]https://competitions.codalab.org/competitions/20011

Details	# Tweets Train Dataset	# Tweets in Test Dataset
Total Posts in Sub-task A	13240	860
Offensive posts	4440	240
Non-offensive posts	8800	620
sub-task-B : Targeted (TIN) posts	3876	213
Non-Targeted (UNT) posts	524	27
Sub-task C: Individual	2407	100
Group	1074	78
Other	395	35

Table 1: Dataset statistics

to be trained from scratch.(Howard and Ruder, 2018) termed this techniques as shallow representation against the hierarchical representation.

The rest of this paper is organized as follows. In section 2 we briefly discuss the related work in this area. In section 3, we present our method and model. In section 4, we present results and give the brief analysis. In section 5, we will give our final conclusion along with future works.

2 Related Work

Hate Speech Detection research attracts researchers from diverse background like computational Linguistic, computer science, and social science. The actual term hate speech was coined by (Warner and Hirschberg, 2012). Various Authors used different notion like offensive language (Razavi et al., 2010), cyberbullying (Xu et al., 2012), aggression (Kumar et al., 2018). (Davidson et al., 2017) studied tweet classification of hate speech and offensive language and defined hate speech as following: language that is used to express hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group. Authors observed that offensive language is often miss-classified as hate speech. They have trained a multi-class classifier on N-gram features weighted by its TF-IDF weights and PoS tags. In addition to these, features like sentiment score of each tweet, number of hashtags, and URLS, mentions are considered. Authors concluded that Logistic Regression and Linear SVM are better than NB, Decision Tree, and Random Forests. (Schmidt and Wiegand, 2017) perform comprehensive survey on hate speech. They have identified features like surface features, sentiment, word generalization, lexical, linguistics etc. can be used by classifier.

(Malmasi and Zampieri, 2018) tried to address the problem of discriminating profanity from hate speech in the social media posts. N-grams, skip-gram and clustering based word representation features are considered for the 3-class classification. The author uses SVM and advance ensemble based classifier for this task and achieved 80% accuracy. (Gambäck and Sikdar, 2017) performed 4-class classification on Twitter messages using CNN with word embedding generated through Word2vec and character n-grams. Authors claim that word embedding generated through Word2vec outperformed random vector and n-gram characters. (Zhang et al., 2018) proposed a new method based on CNN and LSTM with drop out and pooling for hate speech detection. Authors concluded that their method achieved improvement on F1 score of most of the hate speech datasets.

3 Methodology and Data

Since the social media data suffers from the data sparsity problem, classifier based on the BoW features might not be appropriate as compared to distributed word representation. Our previous work (Majumder et al., 2018) also supported this intuition. Empirical evidence (Majumder et al., 2018) suggest that pre-trained vector trained on huge corpus provides better word embedding than embedding generated from a limited training corpus. Some authors (Howard and Ruder, 2018) termed it as a shallow-transfer learning approach. In this method, first layer or embedding layer of deep neural network is initialized with pre-trained vectors and the rest of the network is trained from scratch. Since fastText generates word embedding for a word which is unseen during the training by using the subword or n-gram of the word, it is the better choice than Word2vec and Glove. As discussed in the previous section, there is substantial class imbalance particularly in sub-task B and C. To address this issue, class weights are incorporated into the cost function of the classifier which gives higher weight to minority class and lower weights to the majority class. Four deep learning based models: Bidirectional LSTM, Single LSTM, CNN and stacked CNN are designed for the classification.

Pre-processing : Track organizers have partially pre-processed tweets in the dataset. User mention, URL are replaced with standard tags. We did not perform any sort of further pre-processing or stemming on the texts.

Word embedding : fastText pre-trained word vectors with dimension 300 are used to initialize the embedding layer. This model is trained on 600B tokens of commonly crawled corpus.

3.1 Model Architecture and Hyperparameters

In this sub-section, we briefly describe our models used for the classification. The first model is based on Bidirectional LSTM model includes the embedding layer with 300 dimensions, Bidirectional LSTM layer with 50 memory units followed by one-dimensional global max pooling and dense layer with softmax/sigmoid activations. Hyperparameters are as follows: Sequence length is fixed at 30. Number of features is equal to the half of total vocabulary size in each task. Models are trained for 10 epoch. Adam optimization algorithm is used to update network weights.

The second model is based on LSTM. The model includes embedding layer with 300 dimensions, LSTM layer with 64 memory units, followed by two dense layers with softmax/sigmoid activations. A dropout layer is added to the hidden layer to counter the overfitting. Hyperparameters of the model is the same as the first model.

Rest of the two models are based on Convolution Neural Network, includes embedding layer with embed size of 300, followed by a one-dimensional convolution layer with 100 filters of height 2 and stride 1 to target biagrams. In addition to this, Global Max Pooling layer is added. Pooling layer fetches the maximum value from the filters which are feeded to the dense layer. There are 256 nodes in the hidden layer without any dropout. The last model is same as previous CNN model except three one-dimensional convolution layer are stacked together. Different

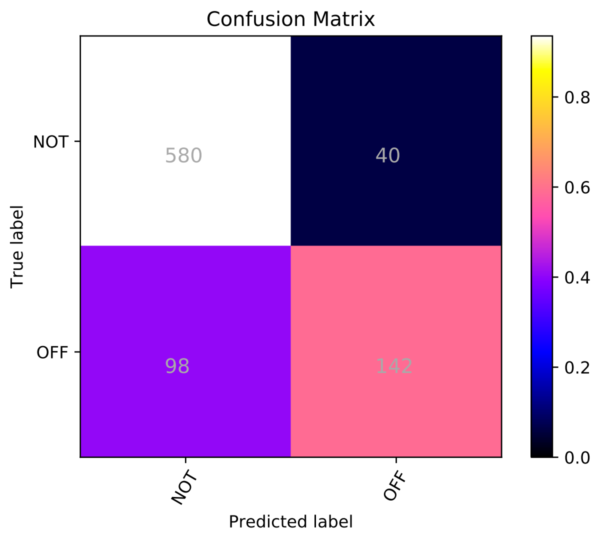


Figure 1: Sub-task A, Confusion Matrix:BLSTM classifier

one-dimensional filters with height 2,3,4 to target bigrams, trigrams, and four-grams features. After convolution layers and max pool layer, model concatenate max pooled results from each of one-dimensional convolution layers, then build one output layer on top of them. (Majumder et al., 2018). Hyperparameters of the model is the same as the first model.

4 Results

In this section, we report the results obtained by the model discussed in the previous section. Table 2, 3, 4 display results of sub-task A, B, and C, respectively. We have randomly splitted the dataset into 80% training and 20% validation. By and large, results on test dataset are better than cross-validation. F1-macro score is the primary metric for the evaluation. Results are comparable with the top team and substantially outperforms all the random baselines. Figure 1, 2, and 3, show the confusion matrices for all the sub-tasks.

5 Conclusion

In this paper, we have presented our deep learning based approach for multi-level offensive text classification. The system reports reasonable performance. Macro f1 and accuracy score around 78.3% and 84% in sub-task A. In sub-task B, our system performs the worst in UNT class(offensive post without target). The reason behind this underperformance is few number of training examples for the UNT class. Similar case happened in the

	Test Dataset		Cross Validation	
System	**F1 (macro)**	**Accuracy**	**F1 (macro)**	**Accuracy**
All NOT baseline	0.4189	0.7209		
All OFF baseline	0.2182	0.2790		
Bidirectional LSTM	**0.7833**	0.8395	0.75	0.79
CNN	0.7800	0.8337	0.76	0.7915
LSTM-balanced	0.7500	0.8047	0.74	0.7708
Top-run: pliu19	0.829			

Table 2: Results for Sub-task A.

	Test Dataset		Cross Validation	
System	**F1 (macro)**	**Accuracy**	**F1 (macro)**	**Accuracy**
All TIN baseline	0.4702	0.8875		
All UNT baseline	0.1011	0.1125		
CNN	**0.6456**	0.9042	0.60	0.8875
LSTM-balanced	0.5471	0.825	0.60	0.867
CNN-balanced	0.6455	0.8917	0.55	0.8943
Top Team jhan014	0.755			

Table 3: Results for Sub-task B

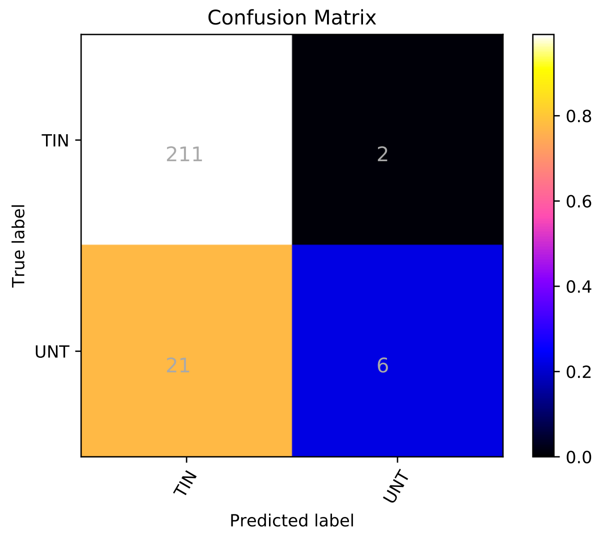


Figure 2: Sub-task B, Confusion matrix : CNN classifier

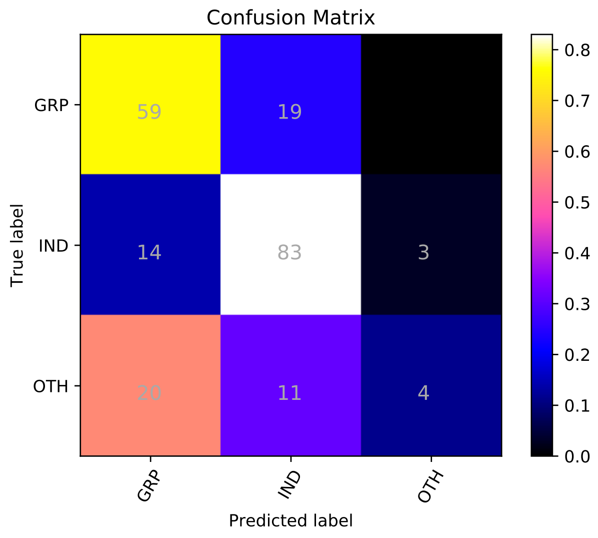


Figure 3: Sub-task C, Confusion Matrix:CNN Classifier

	Test Dataset		Cross Validation	
System	**F1 (macro)**	**Accuracy**	**F1 (macro)**	**Accuracy**
All GRP baseline	0.1787	0.3662		
All IND baseline	0.2130	0.4695		
All OTH baseline	0.0941	0.1643		
CNN-balanced	**0.5533**	0.6854	0.5231	0.695
BLSTM-balanced	0.4829	0.662	0.5223	0.6959
Stacked CNN	0.5198	0.662	0.5074	0.6920
Top Team vradivchev	0.66			

Table 4: Results for Sub-task C

sub-task C. We have set class weights in the cost function of the model. Unfortunately, it did not work. In the future, we will try to address imbalance class problem using external vocabulary augmentation. we would like to explore various transfer learning model like BERT, ELMO and ULMFit for this multi-level classification problem.

References

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Prasenjit Majumder, Thomas Mandl, et al. 2018. Filtering aggression from the multilingual social media feed. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 199–207.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Amir H Razavi, Diana Inkpen, Sasha Uritsky, and Stan Matwin. 2010. Offensive language detection using multi-level classification. In *Canadian Conference on Artificial Intelligence*, pages 16–27. Springer.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26. Association for Computational Linguistics.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from bullying traces in social media. In *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 656–666. Association for Computational Linguistics.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

DeepAnalyzer at SemEval-2019 Task 6: A deep learning-based ensemble method for identifying offensive tweets

Gretel Liz De la Peña Sarracén and **Paolo Rosso**
PRHLT Research Center, Universitat Politècnica de València, Spain

Abstract

This paper describes the system we developed for SemEval 2019 on Identifying and Categorizing Offensive Language in Social Media (OffensEval - Task 6). The task focuses on offensive language in tweets. It is organized into three sub-tasks for offensive language identification; automatic categorization of offense types and offense target identification. The approach for the first subtask is a deep learning-based ensemble method which uses a Bidirectional LSTM Recurrent Neural Network and a Convolutional Neural Network. Additionally we use the information from part-of-speech tagging of tweets for target identification and combine previous results for categorization of offense types.

1 Introduction

The use of Internet has become an important media of personal and commercial communication. In this scenario, some users take advantage of the anonymity of this kind of communication, using this to engage in behaviour that many of them would not consider in real life. Therefore, much of the offensive language is widespread in social networks. Then, studying offensive language in texts from the social media is an essential task for security, the prevention of cyber-bullying, among other abusive behavior.

To increase the research in this areas, several workshops have been organized, such as ALW[1] and TRAC[2]. Recently, OffensEval[3] (Zampieri et al., 2019b), which is a shared task at the SemEval-2019[4] workshop has been launched on the research community. The aim of OffensEval is to deal with offensive language detection in the English language focusing on messages from Twitter.

In OffensEval, the treatment of offensive content is divided into three subtasks taking the type and target of offenses into account:

- A: Offensive language identification.
- B: Automatic categorization of offense types.
- C: Offense target identification.

In this work, we present the methodology proposed to each of these sub-tasks, which includes an ensemble of a LSTM Recurrent Neural Network and a Convolutional Neural Network, and additionally linguistic features for the last two subtasks. The architecture of the system will be more detailed in the following sections.

The paper is organized as follows. Next section briefly describes other works in this area. Then, Section 3 describes the proposed metodology and the dataset. Results are discussed in Section 4. Finally, we draw our conclusions together with a summary of our findings in Section 5.

2 Related Work

Some approaches have been proposed to tackle the problem of offensive language detection. It is the case of recent works (Waseem et al., 2017; ElSherief et al., 2018; Gambäck and Sikdar, 2017; Zhang et al., 2018) and surveys (Schmidt and Wiegand, 2017) and (Fortuna and Nunes, 2018). There are even studies on languages other than English such as (Su et al., 2017) on Chinese and (Fišer et al., 2017) on Slovene.

Many of the last approaches rely on neural network models. For instance, the work of (Ganesan et al., 2018) presents a Multi-Layer Feedforward Neural Networks. Moreover, (Park and Fung, 2017) proposes to implement three models based

[1]https://sites.google.com/site/abusivelanguageworkshop2017/
[2]https://sites.google.com/view/trac1/home
[3]https://competitions.codalab.org/competitions/20011
[4]http://alt.qcri.org/semeval2019/index.php?id=tasks

on Convolutional Neural Networks (CNN) to classify sexist and racist abusive language: Char-CNN, WordCNN, and HybridCNN. It work reports that can boost the performance of simpler models. Also, (Pitsilis et al., 2018) proposes a detection scheme that is an ensemble of Recurrent Neural Network classifiers. It incorporates various features associated with user-related information. They report that the scheme can successfully distinguish racism and sexism messages from normal text.

3 Methodology and Data

The corpus provided by the organizers consists of 14,100 tweets in English. The data collection methods used to compile the dataset used in OffensEval is described in Zampieri et al. (2019a).

The first step is the preprocessing of the tweets, where texts are cleaned. All emoticons, hashtag and urls are removed. Then, the texts are represented as vectors with word embedding vectors. We used the pre-trained word vectors of Glove (Pennington et al., 2014), trained on 2 billion words from Twitter.

The method proposed in this work is based on an architecture that sequentially obtains the output for each of the subtasks. In the first level we use a model whose input is the word embeddings of a tweet and the output is a vector (*r_vector*) that is taken as a compact representation of the input and is used in the following steps. For the model, two types of networks have been used. In a first approach a Recurrent Neural Network (RNN) is used, and as a second approximation a Convolutional Neuronal Network (CNN). These two models are described below.

3.1 Convolutional Neural Network

The model is a version of the convolutional neural networks presented in (Kim, 2014) for sentence-level classification tasks. Here, the input of the model is a matrix where each row corresponds to the embedding vector of each word in the tweet. Three different filters of sizes 3, 4 and 5 are applied in a 1D convolution step to capture information from 3-grams, 4-grams and 5-grams. The feature maps produced by the convolution layer are forwarded to a Maxpooling layer. We used 2x2 filters for this pooling function on a feature map to reduce it to the single most dominant features.

Finally, the r_vector is generated by the concatenation of the results for each of the filters.

3.2 Recurent Neural Network

In NLP problems, standard LSTM Recurrent Neural Networks receive sequentially (left to right order) at each time step a word embedding w_t and produces a hidden state h_t.

On the other hand, the bidirectional LSTM makes the same operations as standard LSTM but, processes the incoming text in a left-to-right and a right-to-left order in parallel. Thus, the output is a two hidden state at each time step $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$. The proposed method uses a Bidirectional LSTM network which considers each new hidden state as the concatenation of these two $\hat{h_t} = [\overrightarrow{h_t}, \overleftarrow{h_t}]$. The idea of this Bi-LSTM is to capture long-range and backwards dependencies.

3.3 Sub-task A

For the first sub-task, which consists in the identification of offensive language in tweets, *r_vector* is used as input of a Fully Connected Neural Networks (FCNN) of two layers with activation function relu. The class (offensive or not) is obtained in a third layer of two units, that refer to the number of classes, with a softmax activation function.

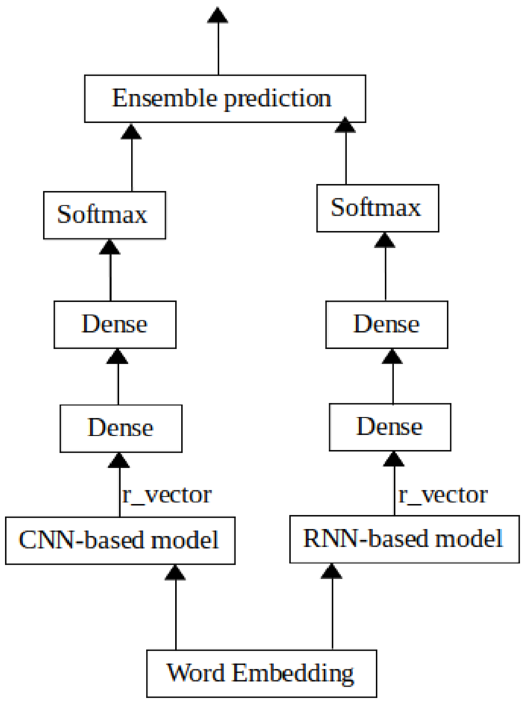


Figure 1: Architecture. Sub-task A

The Figure 1 shows the general scheme commented. Given this architecture, three weights of both CNN and RNN models were made. In the first weighting all the weight is for RNN (RNN_run). In contrast, in the second one, all the weight is for CNN (CNN_run). Finally, the third one is the actual ensemble model where both models are assigned equal weight (Ensemble_run). For combining the results of both models, the system gets the mean of the predictions of each one.

3.4 Sub-tasks B and C

In the sub-task of detecting the target of offensive language, the information of the part-of-speech tagging process of the tweets is used. This allows us to make more fine-grained distinctions on the words in texts which can identify to the target of aggressiveness. For instance, this information allows to discriminate between a proper noun and other kind of noun, and if a noun is plural or singular. In this way the model can learn sequences of tags which represent each type of target. The POS labels are obtained with Standford CoreNLP and they are represented as a one hot vectors. The sequence of labels is analyzed with a LSTM RNN, and a representation *p_vector* is obtained. Then, the concatenation of vectors *r_vector* and *p_vector* is used as input to another FCNN of one hidden layer with the activation function relu, and an output layer with two neurons with a softmax activation function. In this way, the prediction corresponding to the offensive target in the tweets is obtained. The Figure 2 shows this processing.

Finally, for the sub-task of classifying the types of offensive tweet, the prediction is obtained in a similar way to the previous sub-task. Here, a one hot vector corresponding to the POS tags present in the tweet is added to *r_vector*. Then, the prediction is calculated using another FCNN.

Finally, cross entropy is used as the loss function, which is defined as:

$$L = -\sum_{i} y_i * log(\hat{y}) \quad (1)$$

Where y_i is the ground true classification of the tweet and $\hat{y}$ the predicted one.

4 Results

In the evaluation, the official ranking metric is macro-averaged F1. The results obtained in each subtask are shown in the next tables and confusion matrices. For each case, each of the three approaches discussed above (CNN_run, RNN_run

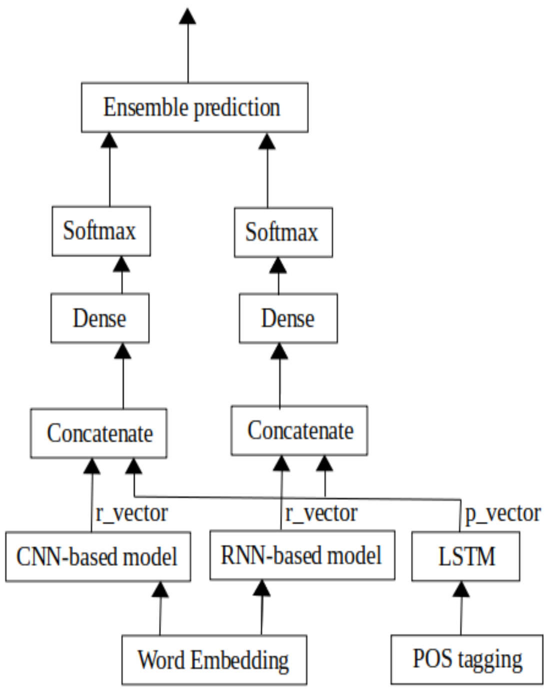


Figure 2: Architecture. Sub-task B

and Ensemble_run) was evaluated and the results are shown in the tables with the name that was indicated. Also, random baseline generated by assigning the same labels for all instances are included. For example, ”All OFF” in sub-task A represents the performance of a system that labels everything as offensive. It was used for comparison.

System	macro F1
All NOT baseline	0.4189
All OFF baseline	0.2182
Best	0.829
RNN_run	0.5984
CNN_run	**0.6600**
Ensemble_run	0.5925

Table 1: Results for Sub-task A

These results reveal a behavior not as good as expected, since although the baselines were exceeded in each case, the results were relatively far from the best results of the competition. Perhaps this is due to the fact that the different linguistic characteristics that could be extracted from tweets, such as information related to emoticons, hashtags and urls, were not analyzed in detail.

Another aspect to note is that for the three tasks

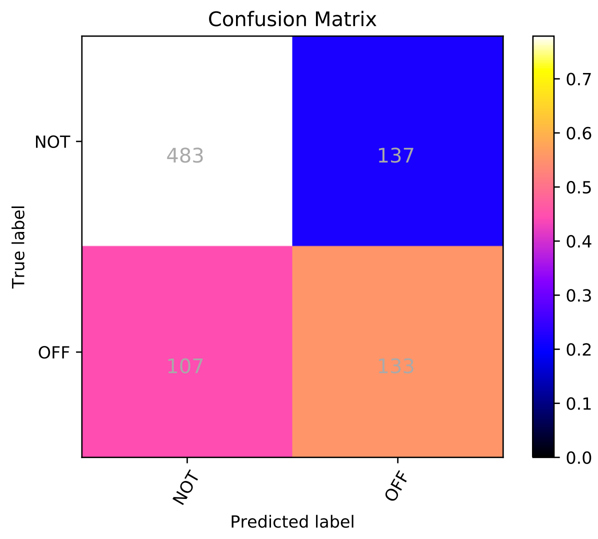


Figure 3: Sub-task A: CNN_run

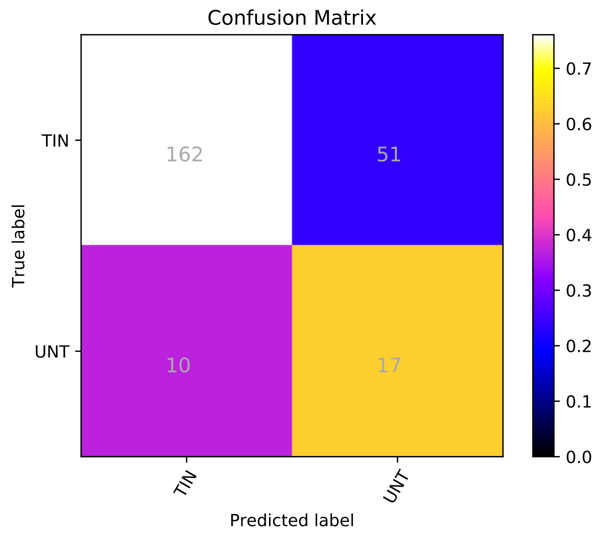


Figure 4: Sub-task B: RNN_run

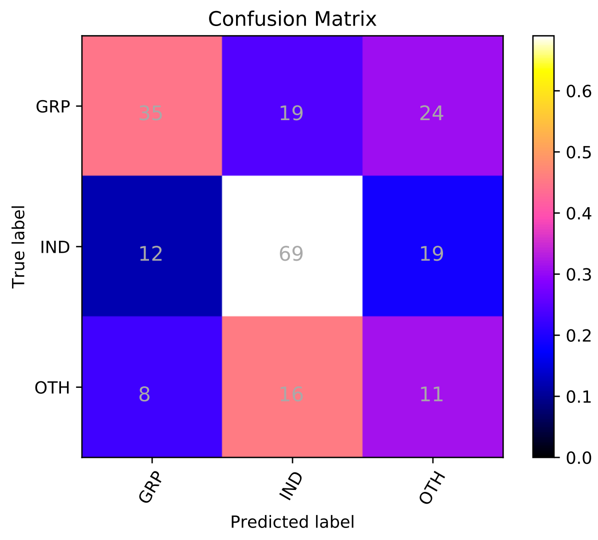


Figure 5: Sub-task C: CNN_run

System	macro F1
All TIN baseline	0.4702
All UNT baseline	0.1011
Best	0.755
RNN_run	**0.5997**
CNN_run	0.5704
Ensemble_run	0.5587

Table 2: Results for Sub-task B.

System	macro F1
All GRP baseline	0.1787
All IND baseline	0.2130
All OTH baseline	0.0941
Best	0.660
RNN_run	0.3848
CNN_run	**0.4833**
Ensemble_run	0.4174

Table 3: Results for Sub-task C.

the best approach is based on simple models instead of a combination of models that in our case was obtained with an ensemble of models based on neural networks. So that, for two of the tasks the best results were obtained only with the use of CNN and for the other one with the RNN.

5 Conclusion

In this paper our solution for the OffensEval challenge in SemEval 2019 was presented. We used an ensemble of models based on deep learning, and compared the results obtained to those ob- tained with each of the models independently. As a conclusion, it can be said that it may be more important for this kind of tasks to search for properly linguistic characteristics instead of designing complex models with a lot of parameters.

References

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Darja Fišer, Tomaž Erjavec, and Nikola Ljubešić. 2017. Legal Framework, Dataset and Annotation Schema for Socially Unacceptable On-line Discourse Practices in Slovene. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Aishwarya Ganesan, Sneha Birajdar, Shivani Dalvi, and Jagruti Dandekar. 2018. Offensive language detection using ai technique.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Ji Ho Park and Pascale Fung. 2017. One-step and two-step classification for abusive language detection on twitter. *arXiv preprint arXiv:1706.01206*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Georgios K Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Huei-Po Su, Chen-Jie Huang, Hao-Tsung Chang, and Chuan-Jie Lin. 2017. Rephrasing Profanity in Chinese Text. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

NLP at SemEval-2019 Task 6: Detecting Offensive language using Neural Networks

Prashant Kapil[*], **Asif Ekbal**[*] **and Dipankar Das**[+]

[*]Department of Computer Science and Engineering
Indian Institute of Technology Patna, India
[+] Department of Computer Science and Engineering
Jadavpur University Kolkata, India
[*]{*prashant.pcs17, asif*}*@iitp.ac.in*
[+]*ddas@cse.jdvu.ac.in*

Abstract

In this paper we built several deep learning architectures to participate in shared task *OffensEval: Identifying and categorizing Offensive language in Social media by semEval-2019* (Zampieri et al., 2019b). The dataset was annotated with three level annotation schemes and task was to detect between offensive and not offensive, categorization and target identification in offensive contents. Deep learning models with POS information as feature were also leveraged for classification. The three best models that performed best on individual sub tasks are stacking of CNN-Bi-LSTM with Attention, BiLSTM with POS information added with word features and Bi-LSTM for third task. Our models achieved a Macro F1 score of 0.7594, 0.5378 and 0.4588 in Task(A,B,C) respectively with rank of 33rd, 54th and 52nd out of 103, 75 and 65 submissions.

1 Introduction

Due to the exponential rise in the usage of internet user generated content in the form of blogs, posts, comments etc. have been increased manifold. Some users also using this platform to target any individual or any particular group on social media on the basis of certain attributes, sharing different views. Many studies have been conducted on offensive language, hate speech, cyberbullying, profanity, aggression detection. These contents are major concern for governments, so robust computational systems need to be developed to tackle these posts to maintain social harmony. This paper is organized as follows. Related work have been discussed in section 2, Methodology have been described in Section 3 followed by data sets and other settings used to solve the tasks in Section 4. Results and analysis of the models is described in Section 5 with the limitation of the models in Error Analysis in section 6. Section 7 contains the conclusion and Future scope.

1.1 Problem Definition

The organizers proposed a hierarchical three level annotation model and divided into three sub tasks.
Task A: This task consist of classifying between *offensive* and *not offensive* comments
Task B: The Offensive language was further needs to be classified into *Targeted(TIN)* and *UnTargeted(UNT)*.
Task C: The targeted offensive needs to be further classified into *Individual(IND)*, *Group(GRP)* and *Other(OTH)*.

2 Related Work

(Nockleby, 2000) defined hate speech as any communication that demean any person or any group on the basis of race, color, gender, ethnicity, sexual orientation, and nationality. (Kowalski et al., 2014) defined cyber aggression as using digital media to intentionally harm another person. (Schmidt and Wiegand, 2017) presents a survey on the existing research in this field and different set of features used in machine learning and Deep learning were discussed. (Silva et al., 2016) proposed and validated sentence structure to detect hate speech and also used this to construct hate speech datasets. They also provided the characteristics study to identify the main targets of hate speech in Twitter and Whisper. They designed two rules i.e *I*<intensity><user intent><hate Target> and <one word> people ex:"black people","maxican" people. (Waseem, 2016) examined the performance of classification based on training performed on amateur and expert annotations. (Ross et al., 2017) concluded that hate speech requires significantly better definitions and guidelines. (Sood et al., 2012) detected profanity by identifying offensive words using list based methods and incorporated edit distance to find similar obscene words. (Davidson et al., 2017) observed that seperating offensive and hate speech is very challenging task. *nigga*, *hoe* , *bitch*, *fag* are very offensive in nature but can be used in different manner. They reported Logistic Regression as their best classifier in detecting approx. 25K Tweets by using N-grams weighted by TF-IDF, POS n-grams and using sentiment score as their features.(Samghabadi et al., 2017) used surface level features like word n-grams and char n-grams, LIWC and SentiWordNet to get the sentiment score as well as some domain related features. (Malmasi and Zampieri, 2017) used char n-grams, word n-grams and word skip grams to get accuracy of 78% on Data set of 14509 tweets classified into 3 classes. (Waseem et al., 2017) tried to capture similarities between different sub tasks. They proposed a typology to differentiate language on the basis of individual or group attack or if the content is explicit or implicit. (Gambäck and Sikdar, 2017) used random vector, word vectors and also concatenated word based CNN and character based CNN to classify 6909 tweets into 4 classes. (Xu et al., 2012) used LDA to find out relevant and useful sentiment in bullying texts. (Zhang et al., 2018) proposed a CNN-GRU based structure which outperformed 6 out of 7 datasets by at most 13 F1 points. They have used surface level features, linguistic features, sentiment features as well as number of misspellings , percentage of capitalisation for SVM. (Aroyehun and Gelbukh, 2018) implemented several neural networks and also found that char n-grams is more superior than word n-grams in NBSVM. They also used data augmentation, pseudo labeling and sentiment score as feature. (Kumar et al., 2018) discuss the task of developing a classifier to discriminate *Overtly*,*Covertly* and *Non Aggressive* text using 15000 annotated social media data in both English and Hindi(in Roman and Devanagri script) as part of TRAC-1. (Badjatiya et al., 2017) experimented with several deep neural architecture and found that it outperformed state of the art word/char n-grams.(Djuric et al., 2015) proposed paragraph to vector for modelling of comments. (Gao and Huang, 2017) discusses the Bi-LSTM with attention mechanism with learning components context improved the classifier performance.

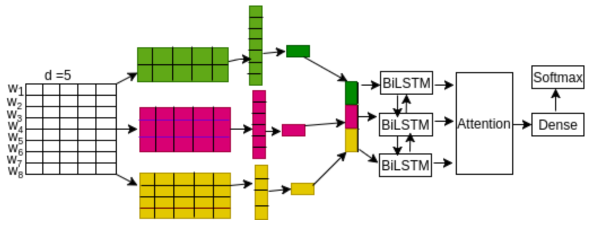


Figure 1: **Sub Task A:CNN-BiLSTM-Attention**

(Founta et al., 2018) studied different forms of abusive behaviour and made public annotated corpus of 80K Tweets categorized into 8 labels like Hate, aggressive, cyber bullying, normal , Spam.

3 Methodology

3.1 Task A:CNN-BiLSTM-Attention

In this model first we converted all the words to their unique index. Then all the unique index in the sentences were mapped to their real valued vectors of Dimensions 100 using Glove by (Pennington et al., 2014) from Embedding Matrix. Convolution layers is used to extract useful information by convolving i words at a time using learnable kernel of size i*h where i = [2,3,4] and h is of size equal to the dimensions. The element wise dot product is performed to get the feature map f_1. N numbers of filters are used to get feature map = [f_1,f_2...f_n]. Pooling reduces the size of representation by selecting max value from each feature map which is then passed to the BiLSTM layer with 100 hidden units. The sentence level representation is then passed to activation layers to capture the important keywords informations. This vector representation is then passed to softmax classification to get the probability values of each class.

Attention It tries to make RNN better by letting the network to know the weight of important keywords. It produces state of the art results on several NLP tasks. We used the approach followed by (Ding et al., 2018) for sentence level attention which follows the following equation.

$$e_t = tanh(Wh_t + b) \quad (1)$$

$$\alpha_t = softmax(e_t) \quad (2)$$

$$output = \sum_{t=1}^{t=n} \alpha_t h_t \quad (3)$$

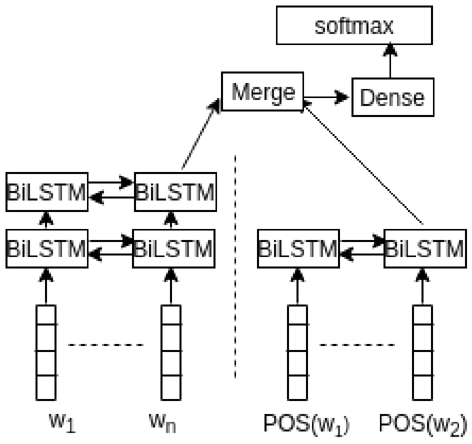


Figure 2: **Sub Task B:BiLSTM(Word+POS)**

3.2 Task B:BiLSTM(Word+POS)

In this model two layers of BiLSTM were used with hidden nodes of 100 where the sentences were being represented by Glove embedding. BiLSTM uses 2 LSTM that is useful for keeping both the past and future information. The input sequence $(i_1,i_2,...i_n)$ is converted to $(h^1_i,h^2_i...h^n_i)$ taking into account each words. Each word was tagged with its POS Tag and embedding for each Tag was calculated. Each sequence was then converted to their POS Tag real valued vector of Dimensions of 20 using embedding matrix. The input sequence is then passed to BiLSTM layer with hidden nodes of 100. The outputs of both the channels were concatenated and passed to the Fully connected layer followed by softmax Classification.

3.3 Task C: BiLSTM

We used BiLSTM using 100 dimensions to represent sequences by fixing the maximum length to 40 . Post padding with 0 was used for shorter sequences as it helps in preserving the information at the borders. After getting desired hidden representation from 2 layers it is passed to the Fully Connected layers followed by softmax Classifier for getting probability distribution among classes.

4 Data Sets

The Datasets provided by organisers (Zampieri et al., 2019a) were three level annotated social media text. The task was divided into three parts,description of their data sets is in Table**1**, Table **2** and Table **3** .

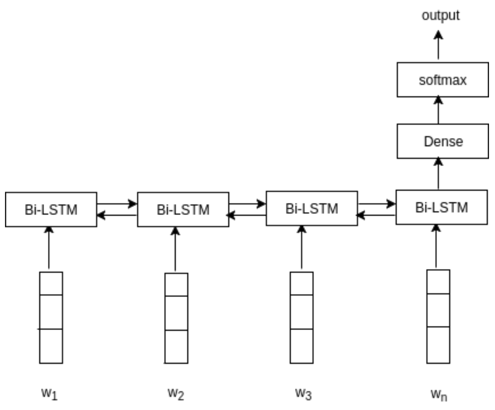


Figure 3: **Sub Task C:BiLSTM**

Class	#Training	#Test
Offensive	4400	240
Not Offensive	8840	820

Table 1: **Data set:Task A**

Class	#Training	#Test
Off_Targeted	3876	213
Off_Untargeted	524	27

Table 2: **Data set: Task B**

Class	#Training	#Test
Off_Tar_IND	2407	100
Off_Tar_GRP	1074	78
Off_Tar_OTH	395	35

Table 3: **Data set:Task C**

4.1 Parameter Tuning,Word embedding and evaluation Metrics

We use Keras with Tensorflow as backend,Scikit-learn library for implementation. For every dataset we use 80:20 for 80% to use in Training and using grid search to learn batch size and epochs. Experiments were performed using stratified 5-fold cross validation to train all the classes according to their proportion and 20% of remaining data were used as testing the model. We are reporting our results on Training data provided by orgainsers by standard Precision, Recall and F-score by averaging all the cross fold results. Categorical cross entropy loss function and Adam optimiser were used for training . In the experiment we use publicly available Glove embedding by (Pennington et al., 2014). We used batch size of(16,32,64) and drop out of (0.1,0.2,0.3).

4.2 Preprocessing

As the datasets are collected from social media it contains lots of noise and inconsistencies in the form of urls, typos and abbreviations. So we start by applying light preprocessing by expanding all appostrophes containing words and then removing characters like : , & ! ? and also all the tokens were tranformed to lower case to avoid capitalized versions of same word being treated as different words. We also used dictionary to expand the misspelled words to its original form. The POS tags were obtained from NLTK.

Class	OFF	NOT	F1	Acc.
OFF	2614	1786	74.96	78.95
NOT	1006	7834		

Table 4: **Cross validation: Task A**

Class	TIN	UNT	F1	Acc.
TIN	3839	37	51.56	88.43
UNT	472	52		

Table 5: **Cross validation: Task B**

Class	IND	GRP	OTH	F1	Acc.
IND	2103	303	1	47.69	71.18
GRP	423	649	2		
OTH	208	182	5		

Table 6: **Cross validation: Task C**

Class	OFF	NOT	F1	Acc.
OFF	131	109	75.94	82.44
NOT	42	578		

Table 7: **Test Set: Task A**

Class	TIN	UNT	F1	Acc.
TIN	212	1	53.78	89.17
UNT	25	2		

Table 8: **Test Set: Task B**

5 Results and Analysis

We have reported the cross validation split accuracy and F-score in Table 4, Table 5 and Table 6 for all the three subtasks. The results for test set is also included in Table 7 .Table 8 and Table 9. For our systems we got almost comparable results for both training and test datasets. We got F-score of 75.94%, 53.78% and 45.88% in sub task A, B, C respectively. Table 10 shows the performance of our system compared with best systems.

Class	GRP	IND	OTH	F1	Acc.
GRP	48	30	0	45.88	64.32
IND	11	89	0		
OTH	15	20	0		

Table 9: **Test Set: Task C**

Task	Ours	Best
A	75.94%	82.9%
B	53.78%	75.5%
C	45.88%	66%

Table 10: **System Performance**

6 Error analysis

Error analysis was carried out to analyze the errors that we encountered in our system by quantitative analysis using Confusion matrix of our best models for each task.

6.1 Quantitative Analysis

From Table 7 it can be seen that false negative rate of offensive class is 45% where as for Not Offensive True Positive rate is 93.22% in Task 1. 42 instances of Not Offensive also got misclassified as Offensive showing evidence of challenges in classification. For Task2 Table 8 shows that TIN True positive rate is almost 100% but system fails to classify UNT class with only 0.08% true positive rate. For Task3 Table 9 shows that system completely fails to detect OTH class with false negative rate of 100%. However GRP and IND class obtained True positive rate of 61.5% and 89% respectively . The misconversion instances of GRP and IND to each other is 30 and 11.

7 Conclusion and Future Scope

In this paper we have explored the effectiveness of deep neural network for Offensive speech detection. We can conclude that fine grained analysis of offensive language detection needs careful attention. Linguistic features can also be leveraged for improvement in classifier.

References

Segun Taofeek Aroyehun and Alexander Gelbukh. 2018. Aggression detection in social media: Using deep neural networks, data augmentation, and pseudo labeling. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 90–97.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Zixiang Ding, Rui Xia, Jianfei Yu, Xiang Li, and Jian Yang. 2018. Densely connected bidirectional lstm with applications to sentence classification. *arXiv preprint arXiv:1802.00889*.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pages 29–30. International World Wide Web Conferences Steering Committee.

Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large Scale Crowdsourcing and Characterization of Twitter Abusive Behavior. *arXiv preprint arXiv:1802.00393*.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Lei Gao and Ruihong Huang. 2017. Detecting online hate speech using context aware models. *arXiv preprint arXiv:1710.07395*.

Robin M Kowalski, Gary W Giumetti, Amber N Schroeder, and Micah R Lattanner. 2014. Bullying in the digital age: A critical review and meta-analysis of cyberbullying research among youth. *Psychological bulletin*, 140(4):1073.

Ritesh Kumar, Atul Kr Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 1–11.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 467–472.

John T Nockleby. 2000. Hate speech. *Encyclopedia of the American constitution*, 3:1277–79.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Wojatzki. 2017. Measuring the reliability of hate speech annotations: The case of the european refugee crisis. *arXiv preprint arXiv:1701.08118*.

Niloofar Safi Samghabadi, Suraj Maharjan, Alan Sprague, Raquel Diaz-Sprague, and Thamar Solorio. 2017. Detecting nastiness in social media. In *Proceedings of the First Workshop on Abusive Language Online*, pages 63–72.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Leandro Silva, Mainack Mondal, Denzil Correa, Fabrício Benevenuto, and Ingmar Weber. 2016. Analyzing the targets of hate in online social media. In *Tenth International AAAI Conference on Web and Social Media*.

Sara Sood, Judd Antin, and Elizabeth Churchill. 2012. Profanity use in online communities. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1481–1490. ACM.

Zeerak Waseem. 2016. Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. In *Proceedings of the first workshop on NLP and computational social science*, pages 138–142.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from bullying traces in social media. In *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 656–666. Association for Computational Linguistics.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. A Hierarchical Annotation of Offensive Posts in Social Media: The Offensive Language Identification Dataset. In *arxiv preprint*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). *arXiv preprint arXiv:1903.08983*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

Duluth at SemEval-2019 Task 6: Lexical Approaches to Identify and Categorize Offensive Tweets

Ted Pedersen
Department of Computer Science
University of Minnesota
Duluth, MN 55812 USA
tpederse@d.umn.edu

Abstract

This paper describes the Duluth systems that participated in SemEval–2019 Task 6, Identifying and Categorizing Offensive Language in Social Media (OffensEval). For the most part these systems took traditional Machine Learning approaches that built classifiers from lexical features found in manually labeled training data. However, our most successful system for classifying a tweet as offensive (or not) was a rule-based black–list approach, and we also experimented with combining the training data from two different but related SemEval tasks. Our best systems in each of the three OffensEval tasks placed in the middle of the comparative evaluation, ranking 57th of 103 in task A, 39th of 75 in task B, and 44th of 65 in task C.

1 Introduction

Social media is notorious for providing a platform for offensive, toxic, and hateful speech. There is a pressing need for NLP tools that can identify and moderate this type of content.

The OffensEval task (Zampieri et al., 2019b) focuses on identifying offensive language in tweets, and determining if specific individuals or groups are being targeted. Our approach was to rely on traditional Machine Learning methods as implemented by Scikit (Pedregosa et al., 2011) to build classifiers from manually labeled examples of offensive tweets. Our models relied on lexical features, including character ngrams, words, and word ngrams. We also included a dictionary based black–list approach, and experimented with combining training data from two related yet different SemEval-2019 tasks.

Offensive language is an umbrella term that covers hate speech, cyber-bullying, abusive language, profanity, and so on. Recognizing offensive language is an important first step in dealing with different kinds of problematic text. Language that is offensive may simply violate community standards regarding the use of profanity, but in other cases may cross over to become abusive, threatening, or dangerous. Detecting such language has proven to be a challenging problem, at least in part because it remains difficult to make distinctions between the casual and even friendly use of profanity versus more serious forms of offensive language (Fortuna and Nunes, 2018; Schmidt and Wiegand, 2017).

2 Experimental Data

OffensEval is made up of three tasks that were carried out in stages during January 2019. Task A is to classify a tweet as being offensive (OFF) or not (NOT). Task B takes the offensive tweets from task A and decides if they are targeted insults (TIN) or not (UNT). Task C looks at the targeted insults from task B and classifies them as being directed against an individual (IND), group (GRP) or other entity (OTH).

Task A provides 13,240 training tweets, of which 8,840 (66.7%) were not offensive (NOT). Task B is made up of the 4,400 training tweets that were offensive (OFF), where 3,876 (88.1%) of these are targeted insults (TIN). Task C includes the targeted insults from task B, of which 2,407 (62.1%) were targeted against individuals (IND) and 1,074 (27.7%) were against groups (GRP). Additional details about the task data can be found in (Zampieri et al., 2019a)

The distribution of classes in the evaluation data was similar. Task A has 860 evaluation tweets of which 620 (72%) were not offensive. Task B includes 240 offensive evaluation tweets, where 213 (89%) were targeted insults. These made up the evaluation data for task C, where 100 (47%) were against individuals, and 78 (37%) were against groups.

The amount of training data is modest, particularly for tasks B and C. In addition, the classes in Task B are quite skewed. Given these factors we decided to rely on traditional Machine Learning techniques, since these have the potential to perform well even with limited training data.

3 System Descriptions

We created three systems for each of the three tasks. Preprocessing was very simple : tweets were converted to lower case and then tokenized into character ngrams, words, or word ngrams.

In task A we relied on unigram models. However, for tasks B and C we focused on character ngrams in order to try and find regularities in the smaller amounts of training data available for these tasks.

3.1 Task A

The goal of Task A was to classify a tweet as offensive (OFF) or not (NOT). We relied on the default settings for vectorization and Machine Learning as provided in Scikit, except as noted below. This results in tokenization based on space separated strings where punctuation and other non-alphanumeric characters are discarded.

A-Sub1 is a Logistic Regression classifier that weighs unigram features using tf-idf.

A-Sub2 is the same as A-Sub1 except that the training data is augmented with the training data from Semeval-2019 Task 5 HatEval (Basile et al., 2019), a task that identifies hate speech. It provides 22,240 training examples where 14,057 (65.2%) are not hate speech. We made the obviously incorrect assumption that tweets that aren't hate speech would also not be offensive. We had hoped that doubling the amount of training data would improve our performance despite our flawed assumption (although it did not).

A-Sub3 is a very simple rule based on a black–list created by merging the following sources:

- words or terms used in offensive tweets five or more times in the OffensEval training data,
- words or terms used in hateful tweets five or more times in the HatEval training data, and
- black–lists found via web search said to be used by WordPress, Google, Facebook, and Youtube[1].

[1]https://www.freewebheaders.com/

Any word or term that appears in two or more of these lists is selected, leading to a master black–list of 563 words. Any tweet that includes any of these words is labeled as offensive (OFF).

3.2 Task B

Task B seeks to identify if an offensive tweet is a targeted insult (TIN) or not (UNT). All three systems relied on character-based ngrams and the default Scikit settings for vectorization and the specific learning method.

B-Sub1 learns a random forest classifier, and B-Sub2 learns a decision tree. Both represent features as 3 alphanumeric character sequences.

B-Sub3 learns a linear Support Vector Machine from the training data. Features are represented as 3 non-space character sequences. We used non-space characters in this case in order to capture special characters that would be discarded by B-Sub1 and B-Sub2. We were particularly interested in retaining # (hashtags) and @ (user ids).

3.3 Task C

All tweets in task C are targeted insults (TIN). The goal is to identify if the target is an individual (IND), group (GRP), or some other entity (OTH).

All of these systems used the default settings from Scikit for vectorization and Machine Learning, except as noted below.

C-Sub1 learns a multinomial Naive Bayesian classifier from the training data. Features are simple unigrams made up of alpa-numeric characters. During development we noticed that Naive Bayes tended to find more balanced distributions of classes than our other approaches.

C-Sub2 learns a decision tree classifier from the training data. Features are 3 character sequences. During development we observed that this was the only method that assigned tweets to the other (OTH) category.

C-Sub3 learns a logistic regression classifier from the training data. Features are word ngrams made up of sequences of 1, 2 and 3 words that occur in more than 10 tweets in the training data.

4 Experimental Results

The official rankings in OffensEval were based on macro–averaged F1, and accuracy was also reported. The performance of individual classes was measured by Precision, Recall, and the $F1$ score.

System	F1-Macro	Accuracy
A-TOP	.83	
A-Sub3	.73	.80
A-Sub2	.69	.80
A-Sub1	.68	.80
A-Baseline	.42	.72
B-TOP	.76	
B-Sub1	.60	.88
B-Sub2	.57	.82
B-Sub3	.52	.77
B-Baseline	.47	.89
C-TOP	.66	
C-Sub1	.48	.67
C-Sub3	.45	.62
C-Sub2	.42	.53
C-Baseline	.21	.47

Table 1: Duluth OffensEval Results

	NOT	OFF		P	R	F1
NOT	609	11	620	.79	.98	.88
OFF	160	80	240	.88	.33	.48
	769	91	860	.82	.80	.77

A-Sub 1: Logistic Regression

	NOT	OFF		P	R	F1
NOT	602	18	620	.80	.97	.88
OFF	152	88	240	.83	.37	.51
	754	106	860	.81	.80	.77

A-Sub 2: Logistic Reg + HatEval

	NOT	OFF		P	R	F1
NOT	558	62	620	.83	.90	.87
OFF	112	128	240	.67	.53	.60
	670	190	860	.79	.80	.79

A-Sub3 : Rule Based, black–list

Table 2: Task A Confusion Matrices

The results of the Duluth Systems are summarized in Table 1. X-TOP is the 1st ranked system in each task. X-Baseline assigns each test tweet to the most frequent class in the training data.

Next, we'll examine the results from each task in more detail. In the confusion matrices provided, the distribution of gold answers (ground truth) is shown on the rows, and the system predictions are on the columns.

4.1 Task A

Task A asks whether a tweet is offensive (or not). It had the largest amount of training data (13,240 examples), of which 33% were considered offensive (OFF) and 67% were not (NOT). In Table 2 and the discussion that follows a true positive is a tweet that is known by ground truth to be not offensive (NOT) and that is predicted to be not offensive (NOT). A true negative is a tweet that is known to be offensive (OFF) and is predicted to be offensive (OFF).

4.2 Confusion Matrix Analysis

A-Sub3 had a modest advantage over the other two systems. A-Sub3 was a simple rule-based black–list approach, while A-Sub1 and A-Sub2 used Machine Learning. All three systems scored identical accuracy (80%), but in looking at their confusion matrices some interesting differences emerge.

Table 2 shows that the rule based method A-Sub3 has a much smaller number of false negatives (112 versus 160 and 152). It also has a larger number of true negatives (128 versus 80 and 88). Overall the rule based system finds more tweets offensive (190) than the Machine Learning methods (91 and 106). This happens because our rule based system only needs to find a single occurrence of one of our 563 black–listed terms to consider a tweet offensive, no doubt leading to many non-offensive tweets being considered offensive (62 versus 11 and 18).

The only difference between A-Sub1 and A-Sub2 was that A-Sub2 had approximately double the number of training tweets. The extra tweets were from the SemEval-2019 hate speech task (HatEval). We hypothesized that more training data might help improve the results of a logistic regression classifier (which was used for both A-Sub1 and A-Sub2). After increasing the training data, A-Sub2 is able to classify exactly one more tweet correctly (690 versus 689). We were somewhat surprised at this very limited effect, although the HateEval corpus is focused on a particular domain of hate speech where the targets are women and immigrants. This does not appear to have matched well with the OffensEval training data.

NOT offensive	OFFensive
user	**SHIT**
antifa	**FUCK**
url	**BITCH**
best	**STUPID**
thank	**ASS**
conservatives	**FUCKING**
new	**IDIOT**
beautiful	**LIAR**
here	**DISGUSTING**
brexit	**SUCKS**
thanks	**SICK**
love	**CRAP**
she	**RACIST**
day	**DUMB**
awesome	**FASCIST**
adorable	NIGGA
safe	FUCKED
voting	**CRAZY**
funny	**IGNORANT**
stand	FOOL
justice	COWARD
idea	IDIOTS
there	SUCK
right	**KILL**
join	PUSSY
well	UGLY
amazing	WORST
twitter	DAMN
welcome	BULLSHIT
trying	ASSHOLE

Table 3: Task A Feature Analysis - A-Sub1

NOT offensive	OFFensive
https	**BITCH**
co	BUILDTHATWALL
immigrant	**SHIT**
men	WOMENSUCK
antifa	**FUCK**
url	**ASS**
user	ILLEGAL
ram	BITCHES
thank	**SUCKS**
best	NODACA
new	BUILDTHEWALL
conservatives	**STUPID**
when	**LIAR**
son	**IDIOT**
you	**DISGUSTING**
stand	WHORE
beautiful	**FUCKING**
kunt	HOE
love	SUCK
justice	ILLEGALS
facebook	**SICK**
ho	**FASCIST**
tonight	**CRAP**
thanks	**IGNORANT**
wondering	THESE
day	**RACIST**
accused	**KILL**
brexit	**CRAZY**
alone	**DUMB**
twitter	WHITE

Table 4: Task A Feature Analysis - A-Sub2

4.2.1 Feature Analysis

Table 3 shows the top 30 most heavily weighted features according to the A-Sub1 logistic regression classifier (which was trained on 13,240 instances). We will have the convention of upper casing features indicative of an offensive tweet and lower casing not offensive features. There are some stark differences between these feature sets, where the offensive ones are for the most part profanity and insults.

In Table 4 we show the top 30 weighted features in A-Sub2, a logistic regression classifier trained on the combination of OffensEval and HatEval data. Terms relating to hatred of women and immigrants abound, and include numerous hash tags (recall that our tokenization only used alphanumerics so # are omitted).

In Tables 3 and 4 we bold face the 18 features that were shared between A-Sub1 and A-Sub2. This gives us some insight into the impact of merging the OffensEval and HatEval training data. Some generic offensive features remain in Table 4 but are strongly augmented by HatEval features that are oriented against women and immigrants.

The 13 shared terms that were indicative of the not offensive class are shown in italics. Some features are what we'd expect for non-offensive tweets : *love, beautiful, thanks, thank, justice* and *best*. Others are more artifacts of the data, *user* is an anonymized twitter id and *url* is an anonymized web site. Others are less clearly not offensive, and seem related to political conversation : *antifa, conservatives,* and *brexit*.

Ratio	Feature	OFF	NOT
61.0	BITCH	61	0
17.5	IDIOT	35	2
14.0	ASSHOLE	14	0
10.6	FUCK	106	10
10.2	STUPID	92	9
10.0	DICK	10	1
10.0	BITCHES	10	0
9.0	SHIT	278	31
9.0	RAPIST	9	1
7.3	FUCKED	22	3
6.3	FUCKING	82	13
5.8	SUCKS	35	6
5.5	CRAP	33	6
5.3	IDIOTS	16	3
5.0	SCUM	10	2
5.0	MORON	10	2
4.9	ASS	108	22
4.8	IGNORANT	19	4
4.5	LOSER	9	2
4.3	SHITTY	13	3
4.2	BUTT	17	4
4.0	UGLY	12	3
3.8	DUMB	23	6
3.2	PUSSY	13	4
3.2	NIGGA	16	5
3.0	PORN	9	3
2.9	HELL	38	13
2.9	BULLSHIT	23	8
2.6	SUCK	21	8
2.5	KILL	32	13

Table 5: Task A Feature Analysis - A-Sub3

However, there are some inconsistencies to note. In Table 3 NIGGA is not necessarily an offensive term and points to the need for annotators to have subtle understandings of culture (Waseem et al., 2018). In Table 4 *kunt* is a deliberate misspelling meant to disguise intent (c.f. (Gröndahl et al., 2018)).

Table 5 shows the top 30 terms from our black–list system A-Sub3 that proved to be most discriminating in identifying an offensive tweet. Recall that A-Sub3 had the highest F1-Macro score of our task A systems. The first column shows a simple ratio of the number of times a feature is used in an offensive tweet (OFF in 3rd column) versus a not offensive one (NOT in 4th column). The most discriminating feature BITCH occurred in 61 offensive tweets and in 0 that were not offensive.

	TIN	UNT		P	R	F1
TIN	206	7	213	.90	.97	.93
UNT	22	5	27	.42	.19	.26
	228	12	240	.85	.88	.86

B-Sub1 : Random Forest

	TIN	UNT		P	R	F1
TIN	188	25	213	.90	.88	.89
UNT	20	7	27	.21	.26	.24
	208	7 32	240	.83	.81	.82

B-Sub2 : Decision Tree

	TIN	UNT		P	R	F1
TIN	179	34	213	.90	.84	.87
UNT	21	6	27	.15	.22	.18
	200	40	240	.81	.77	.79

B-Sub3 : Linear SVM

Table 6: Task B Duluth Systems

4.3 Task B

Task B includes 4,400 training tweets, all of which are judged by ground truth to be offensive. This is a fairly modest amount of training data, particularly given how noisy tweets tend to be. As a result we shifted from using unigrams as features (as in Task A) and moved to the use of character ngrams, in the hopes of identifying patterns that may not exist at the unigram level.

The data in Task B is also the most skewed of all the tasks. Nearly 90% of the tweets belonged to the class of targeted insult (TIN). Our three Task B systems used different Machine Learning classifiers, and all tended to produce very skewed results, where most tweets were judged to be targeted insults (TIN). This is clearly illustrated in Table 6, which shows that the random forest classifier (B-Sub1) was better in terms of Precision and Recall for TIN, whereas all three classifiers struggled with the UNT class.

4.4 Task C

Task C had an even smaller amount of training data (3,876 instances). Given a targeted insult, systems were asked to decide if the target an individual (IND), group (GRP) or other (OTHER). These appear as I, G, and O in Table 7. The Other

	G	I	O		P	R	F1
G	53	25	0	78	.70	.68	.68
I	9	90	1	100	.66	.90	.76
O	14	21	0	35	.00	.00	.00
	76	136	1	213	.57	.67	.61

C-Sub1 : Multinomial Naive Bayes

	G	I	O		P	R	F1
G	39	27	12	78	.51	.50	.50
I	21	70	9	100	.63	.70	.66
O	17	15	3	35	.13	.09	.10
	77	112	24	213	.50	.53	.51

C-Sub2 : Decision Tree

	G	I	O		P	R	F1
G	48	25	5	78	.61	.62	.61
I	13	85	2	100	.67	.85	.75
O	18	17	0	35	.00	.00	.00
	79	127	7	213	.54	.62	.58

C-Sub3 : Logistic Regression

Table 7: Task C Duluth Systems

class is very sparse, and C-Sub1 and C-Sub3 did very poorly on it. However, C-Sub2 (a decision tree) had slightly more success. C-Sub1 and C-Sub2 rely on character ngrams, while C-Sub3 uses word unigrams, bigrams, and trigrams as features.

5 Qualitative Review of Training Data

Finally, we qualitatively studied some of the training data for task A and saw that there is potential for some noise in the gold standard labeling. We found various tweets labeled as offensive that seemed innocuous:

- She should ask a few native Americans what their take on this is.
- gun control! That is all these kids are asking for!
- Tbh these days i just don't like people in general i just don't connect with people these days just a annoyance..
- Dont believe the hype.
- Ouch!
- Then your gonna get bitten
- there is no need to be like That

We also found tweets labeled as not offensive despite the presence of insults and profanity.

- Ppl who say I'm not racist are racist. You Are A Racist. Repeat after me
- I'M SO FUCKING READY
- Great news! Old moonbeam Just went into a coma!
- No fucking way he said this!
- Yep Antifa are literally Hitler.
- Any updates re ending your blatant #racism as #Windrush #Grenfell proves you are

The annotation guidelines from the OffensEval organizers seem relatively clear in stating that all profanity should be considered offensive, although an annotator may intuitively wish to make a more nuanced distinction. Resolving these kinds of inconsistencies seems important since the data from task A is also used for task B and C, and so there is a danger of unintended downstream impacts.

6 Conclusion

Offensive language can take many forms, and some words are offensive in one context but not another. As we observed, profanity was often very indicative of offensive language, but of course can be used in much more casual and friendly contexts. This quickly exposes the limits of black–listing, since once a word is on a black–list it use will most likely always be considered offensive. Identifying targeted targeted individuals or organizations using lexical features and Machine Learning was extremely difficult, particularly given the small amounts of training data. Incorporating the use of syntactic analysis and named entity recognition seem necessary to make progress.

We also encountered the challenging impact of domain differences in identifying offensive language. Our attempt to (naively) increase the amount of available training data by combining the OffensEval and HatEval data had no impact on our results, and our feature analysis made it clear that the two corpora were different to the point of not really providing much shared information that could be leveraged. That said, we intend to explore more sophisticated approaches to transfer learning (e.g., (Karan and Šnajder, 2018; Park and Fung, 2017; Waseem et al., 2018)) since there are quite a few distinct corpora where various forms of hate speech have been annotated.

References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Comput. Surv.*, 51(4):85:1–85:30.

Tommi Gröndahl, Luca Pajola, Mika Juuti, Mauro Conti, and N Asokan. 2018. All you need is ïove: Evading hate speech detection. In *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*, pages 2–12. ACM.

Mladen Karan and Jan Šnajder. 2018. Cross-domain detection of abusive language online. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 132–137.

Ji Ho Park and Pascale Fung. 2017. One-step and two-step classification for abusive language detection on twitter. In *Proceedings of the First Workshop on Abusive Language Online*, pages 41–45.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia, Spain. Association for Computational Linguistics.

Zeerak Waseem, James Thorne, and Joachim Bingel. 2018. Bridging the gaps: Multi task learning for domain transfer of hate speech detection. In *Online Harassment*, pages 29–55. Springer.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Emad at SemEval-2019 Task 6: Offensive Language Identification using Traditional Machine Learning and Deep Learning approaches

Emad Kebriaei[1], Samaneh Karimi[2], Nazanin Sabri[3], Azadeh Shakery[4]
[1234]School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Iran
[4]School of Computer Science and Institute for Research in Fundamental Sciences (IPM)
{emad.kebriaei,samanekarimi,nazaninsabri,shakery}@ut.ac.ir

Abstract

In this paper, the used methods and the results obtained by our team, entitled Emad, on the OffensEval 2019 shared task organized at SemEval 2019 are presented. The OffensEval shared task includes three sub-tasks namely Offensive language identification, Automatic categorization of offense types and Offense target identification. We participated in sub-task A and tried various methods including traditional machine learning methods, deep learning methods and also a combination of the first two sets of methods. We also proposed a data augmentation method using word embedding to improve the performance of our methods. The results show that the augmentation approach outperforms other methods in terms of macro-f1.

1 Introduction

With the growth of social networking platforms, the need for automatic methods that manages the emerging issues or facilitate using them is rising. One of the rising trends in social networks such as Twitter is offensive behavior that can cause the offended users leave their social network. Therefore, the need for effective automatic methods for identifying offensive language in textual data is important.

The OffensEval shared task has been organized in order to give a boost to computational methods for identifying and categorizing offensive content on social media. Three sub-tasks defined in the offensEval shared task are identification of offensive language(sub-task A), categorization of offense types(sub-task B) and identification of the offense target(sub-task C) (Zampieri et al., 2019b).

The main goal in sub-task A is to identify offensive tweets from non-offensive ones. By definition, a post is labeled as offensive if it contains any form of non-acceptable language (profanity) or a targeted offense, which can be veiled or direct.

This year, we participated in sub-task A. Our methods for this sub-task include two approaches. In the first approach, traditional machine learning methods, deep learning methods and also a combination method are employed for the task. In the second approach, a data augmentation method is proposed to improve the performance of the methods of the first approach.

2 Related Work

Offenseive language identification which is also known as aggression, cyberbullying, hate speech and abusive language has been widely studied in previous works(Davidson et al., 2017; Malmasi and Zampieri, 2017, 2018).

Based on a survey conducted by Fortuna and Nunes (2018), the majority of previous works are on English and the researchers mainly use machine learning for this task and most proposed methods on abusive content detection have modeled the problem as a binary classification task. Based on another survey (Schmidt and Wiegand, 2017), different types of features have been employed by previous works including surface features, word generalization features such as word embeddings, sentiment-based features, lexical features, linguistic features, knowledge-based features and multimodal information features. The methods utilized for offensive language identification are mainly supervised learning methods including SVM, Random Forest, Naive Bayes and also deep learning approaches (Gambäck and Sikdar, 2017) As an example, (Gambäck and Sikdar, 2017) proposed a model based on convolutional neural networks which takes word embedding vectors of a document as input and decides whether the document contains hate-speech content or not.

3 Methodology and Data

3.1 Datasets

The dataset used in this competition is available as part of the OffensEval 2019 Shared Task Zampieri et al. (2019a). The training set contains 13240 tweets and the test set contains 860 tweets. We have also employed two external datasets including TRAC-1 data (Kumar et al., 2018) and 50K tweets collected by (Founta et al., 2018) in our experiments.

3.2 Features

In our methods, we make use of the following features:

Content-based Features Tweet text contains words which are the most prominent features to convey feelings. Therefore, based on the content of each tweet, we extract the following features as content-based features: the number of mentions, the number of links, the number of hashtags, the average word length, the number of punctuation marks, the average sentence length(based on the number of words in a each sentence), the total number of words, the number of uppercase and the number of emoticons in each tweet.

Sentiment-based Features Usually hate speech has negative sentiment. Thus, using the sentiment information of tweets may improve the performance of our methods. We use three types of sentiment-based features including polarity, subjectivity and emotion. In order to find the emotion label, we trained a random forest classifier on an external dataset annotated for emotions and polarity which contains 40K tweets and 13 classes of emotion (such as happiness, sadness, and anger)[1].

TF-IDF Features TF-IDF is one of the most popular term-weighting approaches which shows the importance of a term in a document or a collection. We use this feature in combination with other features.

Hate-based Feature Hate-based dictionary is a lexicon that can be used to identify hate speech and offensive language (Davidson et al., 2017). We considered the number of hate words and the number of hate n-grams of length 1 to 4 as hate-based features. Hate-base lexicon is available at `www.hatebase.org`.

[1]`https://www.crowdflower.com/`

3.3 Methods

In this section, the methods employed by our team for sub-task A are explained. We used several methods including traditional machine learning methods such as SVM, Random Forest and Naive Bayes in additions to a deep learning method and a combination method. In addition to the methods mentioned above, we proposed an augmentation method in order to improve the performance of our methods.

3.3.1 Traditional Machine Learning Methods

Traditional machine learning methods, in particular, supervised classification methods is known as the most effective approach for offensive language identification. Therefore, in our experiments we applied three classifiers including SVM, Naive-Bayes and Random Forest. Among the most recent methods in the literature, deep learning methods has shown to be an effective approach for offensive language detection. Hence, we employed CNN, as our deep learning solution.

3.3.2 Combination Method

In this method, we employed majority voting rule to combine the results of our best performing systems on the training set. Precisely speaking, for each tweet we find the majority label of three systems which are SVM, CNN trained on over 50k + 13k tweets and another CNN which trained on 50k + 13k + 10k tweets. The results are shown in Table 1.

CNN Architecture: The word-level CNN model has 1D convolution layer with 150 filters and kernel size 6, dropout 0.2, cross entropy loss funtion and four dense layers with ReLU, tanh, sigmoid and softmax activation respectively.

3.3.3 Data Augmentation Method

A common technique to enhance model generalization is data augmentation. In this method, we employed an external dataset containing 50K tweets labeled as hateful, aggressive, normal and spam, in two different ways as follows. In direct augmentation, we added all tweets to the training set such that the tweets labeled as hateful or aggressive are added as offensive and normal or spam labeled tweets as non-offensive.

In indirect augmentation, first of all, the average word embedding of each tweet in the training set is calculated. Then, the average of the embedding vectors in each class is calculated to be used as the representative (or center) of offensive and non-offensive class. Finally, the average word embedding vector of each tweet in the external dataset is calculated and compared with the offensive and non-offensive representative vectors through cosine similarity computation between each tweet and two centers. We defined a threshold for labeling new tweets. If the absolute difference of the distances between tweet's vector and each of the class center is higher than the threshold, we assign tweet to the nearest class. Thus, the tweets of the external dataset are labeled as their most similar class and added to the training set. During the indirect augmentation process, we used word2vec pre-trained Google News model (GoogleNews-vectors-negative300) to calculate embedding vectors of tweets. The threshold is determined 0.03 by experiments.

4 Results

4.1 Models' Performance Evaluation

In this section, the performance of all methods explained in section 3.3 on the training set using 5-fold cross validation is reported. The results of all of the used models on the training set are shown in Table 1. According to table 1, SVM outperforms other two methods in terms of macro-F1. Comparing the results of SVM and CNN shows that these two methods have close performance on the training set.

System	F1 (macro)	Accuracy
Naive Bayes	0.54 (+/- 0.02)	0.70 (+/- 0.02)
Random Forest	0.64 (+/- 0.02)	0.74 (+/- 0.01)
SVM	0.68 (+/- 0.01)	0.73 (+/- 0.01)
CNN	0.67	0.74

Table 1: Results for all methods on the training set using 5-fold cross validation (the variance of the scores for each fold are shown in parentheses)

4.2 Features' Evaluation

In this section, the impact of using the features explained in section 3.2 on the performance of the SVM method is studied. The results are noted in Table 2.

We perform 5-fold cross-validation on the training set and report the results for SVM using different combinations of the feature sets. The first observation is that TF-IDF features outperform other three sets of features in the first section of table 2 which corresponds to using only one feature set. The combination of TFIDF and sentiment-based features, TFIDF and hate-based features and TFIDF, content-based and hate-based features equally show the best performance among all combinations.

4.3 Augmentation Method Evaluation

In this section, the impact of the augmentation method on the performance of our classifier is evaluated. Table 3 shows the results of SVM on the training set using 5-fold cross validation in three different settings; when no augmentation is done, when the external dataset is used directly and when the augmentation method (i.e. using the external data indirectly) is employed. As table 3 shows, the augmentation method produces the best results.

4.4 Results on the Test Set

In this section, the results of three systems that we submitted to OffensEval 2019 is reported. Table 4 shows the results of SVM using augmentation method with two external datasets (SVM-50k+13k), CNN using augmentation method with three external datasets (CNN-50k+13k+10k) and the majority voting method using the outputs of two mentioned methods on the test set. Furthermore, the results of two random baseline generated by assigning the same labels for all instances (all offensive and all non-offensive) are reported for comparison. According to the table, the combination of first two method using majority voting has the best performance.

System	F1 (macro)	Accuracy
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
SVM-50k+13k	0.7076	0.7884
CNN-50k+13k+10k	0.7155	0.7814
Majority Vote	0.7325	0.8186

Table 4: Results for Sub-task A.

5 Conclusion

In this paper, we address the challenge of automatically detecting offensive and non-offensive language in textual content spread in twitter. We conducted experiments with SVM with varying feature sets and CNN model. We also proposed an augmentation method to improve the performance our classifiers.

System	**F1 (macro)**	**Accuracy**
TF-IDF features	0.68 (+/- 0.02)	0.74 (+/- 0.01)
Content features	0.42 (+/- 0.01)	0.73 (+/- 0.02)
Sentiment features	0.50 (+/- 0.01)	0.69 (+/- 0.01)
Hatebased features	0.49 (+/- 0.01)	0.69 (+/- 0.01)
TF-IDF + Content	0.67 (+/- 0.03)	0.73 (+/- 0.02)
TF-IDF + Sentiment	0.71 (+/- 0.02)	0.76 (+/- 0.01)
TF-IDF + Hatebased	0.71 (+/- 0.02)	0.76 (+/- 0.01)
Content + Sentiment	0.54 (+/- 0.02)	0.68 (+/- 0.01)
Content + Hatebased	0.49 (+/- 0.01)	0.68 (+/- 0.01)
Sentiment + Hatebased	0.51 (+/- 0.04)	0.70 (+/- 0.02)
TF-IDF + Content + Sentiment	0.68 (+/- 0.02)	0.73 (+/- 0.01)
TF-IDF + Content + Hatebased	0.68 (+/- 0.03)	0.74 (+/- 0.02)
TF-IDF + Content + Hatebased	0.71 (+/- 0.02)	0.76 (+/- 0.01)
Content + Sentiment + Hatebased	0.56 (+/- 0.02)	0.70 (+/- 0.01)
TF-IDF + Content + Sentiment + Hatebased	0.68 (+/- 0.01)	0.73 (+/- 0.01)

Table 2: Results for SVM method using different sets of features on the training set using 5-fold cross validation (the variance of the scores for each fold are shown in parentheses)

System	**F1 (macro)**	**Accuracy**
Without using the external dataset	0.68	0.73
With using the external dataset directly	0.73	0.89
With using the external dataset indirectly (augmentation method)	0.76	0.88

Table 3: Results for SVM method on the training set without using the external dataset, with using the external dataset directly and with using the external dataset indirectly(the augmentation method)

References

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large Scale Crowdsourcing and Characterization of Twitter Abusive Behavior. *arXiv preprint arXiv:1802.00393*.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 467–472.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the type and target of offensive posts in social media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Embeddia at SemEval-2019 Task 6: Detecting Hate with Neural Network and Transfer Learning Approaches

Andraž Pelicon, Matej Martinc, Petra Kralj Novak
Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
{Andraz.Pelicon, Matej.Martinc, Petra.Kralj.Novak}@ijs.si

Abstract

SemEval-2019 Task 6 was OffensEval: Identifying and Categorizing Offensive Language in Social Media. The task was further divided into three sub-tasks: offensive language identification, automatic categorization of offense types, and offense target identification. In this paper, we present the approaches used by the Embeddia team, who qualified as fourth, eighteenth and fifth on the three sub-tasks. A different model was trained for each sub-task. For the first sub-task, we used a BERT model fine-tuned on the provided dataset, while for the second and third tasks we developed a custom neural network architecture which combines bag-of-words features and automatically generated sequence-based features. Our results show that combining automatically and manually crafted features fed into a neural architecture outperform transfer learning approach on more unbalanced datasets.

1 Introduction

Over the years, computer-mediated communication, like the one on social media, has become one of the key ways people communicate and share opinions. Computer-mediated communication differs in many ways, both technically and culturally, from more traditional communication technologies (Kiesler et al., 1984). However, the ability to fully or partially hide our identity behind an internet persona leads people to type things they would never say to someone's face (Shaw, 2011). Not only is hate speech more likely to happen on the Internet, where anonymity is easily obtained and speakers are psychologically distant from their audience, but its online nature also gives it a far-reaching and determinative impact (Shaw, 2011). Although most forms of intolerance are not criminal, hate speech and other speech acts designed to harass and intimidate (rather than merely express criticism or dissent), deteriorate public discourse and opinions, which can lead to a more radicalized society.

Online communities, social media platforms, and technology companies have been investing heavily in ways to cope with offensive language to prevent abusive behavior in social media. Social media companies Facebook, Twitter and Google's YouTube have greatly accelerated their removal of online hate speech, and report reviewing over two-thirds of complaints within 24 hours. It has been proven in practice that naive word filtering systems do not manage to scale well to different forms of hate and aggression (Schmidt and Wiegand, 2017). The most promising strategy for detecting abusive language is to use advanced computational methods. This topic has attracted significant attention in recent years as evidenced in recent publications (Waseem et al., 2017; Davidson et al., 2017; Malmasi and Zampieri, 2018).

The SemEval-2019 Task 6 — OffensEval: Identifying and Categorizing Offensive Language in Social Media (Zampieri et al., 2019b) is to use machine learning text classification methods to identify offensive content and hate speech. The task organizers have provided a new dataset (Zampieri et al., 2019a) comprised of Twitter posts which employs a three-level hierarchical labeling scheme, according to the three hierarchically posed sub-tasks, where each sub-task serves as a stepping stone for the next sub-task. Sub-task A aims to identify offensive content, Sub-task B aims to classify offensive content as a targeted or untargeted offense, while Sub-task C aims to identify the target of the offense.

In this paper, we present the approaches used by the Embeddia team to tackle the three sub-tasks of SemEval-2019 Task 6: OffensEval. The Embeddia team qualified as fourth, eighteenth and fifth on Sub-tasks A, B and C, respectively. The Embeddia team used different neural architectures and transfer learning techniques (Devlin et al., 2018). We also explore if combining automatically generated sequence-based features with more traditional manual feature engineering techniques improves the classification performance and how different classifiers perform on unbalanced datasets. Our results show that a combination of automatically and manually crafted features fed into a neural architecture outperforms the transfer learning approach on the more unbalanced datasets of Subtasks B and C.

This paper is organized as follows. In Section 2, we present related work in the area of offensive and hate speech detection. Section 3 describes in more detail the provided dataset and the methodology used for the task. Section 4 reviews the results we obtained on the three sub-tasks with our models. Section 5 concludes the paper and presents some ideas for future work.

2 Related Work

A number of workshops that dealt with offensive content, hate speech and aggression were organized in the past several years, which points to the increasing interest in the field. Due to important contributions of publications from TA-COS[1], Abusive Language Online[2], and TRAC[3], hate speech detection became better understood and established as a hard problem. The report on shared task from the TRAC workshop (Kumar et al., 2018) shows that of 45 systems trying to identify hateful content in English and Hindi Facebook posts, the best-performing ones achieved weighted macro-averaged F-scores of just over 0.6.

Schmidt and Wiegand (2017) note in their survey that supervised learning approaches are predominantly used for hate speech detection. Among those, the most widespread are support vector machines (SVM) and recurrent neural networks, which are emerging in recent times (Pavlopoulos et al., 2017). Zhang et al. (2018) devised a neural network architecture combining convolutional and gated recurrent layers for detecting hate speech, achieving state-of-the-art performance on several Twitter datasets. Malmasi and Zampieri (2018) used SVMs with different surface-level features, such as surface n-grams, word skip-grams and word representation n-grams induced with Brown clustering. They concluded that surface n-grams perform well for hate speech detection but also noted that these features might not be enough to discriminate between profanity and hate speech with high accuracy and that deeper linguistic features might be required for this scenario.

A common difficulty that arises with supervised approaches for hate speech and aggression detection is a skewed class distribution in datasets. Davidson et al. (2017) note that in the dataset used in the study only 5% of tweets were labeled as hate speech. To counteract this, datasets are often resampled with different techniques to improve on the predictive power of the systems over all classes. Aroyehun and Gelbukh (2018) increased the size of the used dataset by translating examples to four different languages, namely French, Spanish, German, and Hindi, and translating them back to English. Their system placed first in the Aggression Detection in Social Media Shared Task of the aforementioned TRAC workshop.

A recently emerging technique in the field of natural language processing (NLP) is the employment of transfer learning (Howard and Ruder, 2018; Devlin et al., 2018). The main idea of these approaches is to pretrain a neural language model on large general corpora and then fine-tune this model for a task at hand by adding an additional task-specific layer on top of the language model and train it for a couple of additional epochs. A recent model called Bidirectional encoder representations from transformers (BERT) (Devlin et al., 2018) was pretrained on the concatenation of BooksCorpus (800M words) (Zhu et al., 2015) and English Wikipedia (2,500M words) and then successfully applied to a number of NLP tasks without changing its core architecture and with relatively inexpensive fine-tuning for each specific task. According to our knowledge, it has not been applied on a hate speech detection task yet, however it reached state-of-the-art results in the question answering task on the SQuAD dataset (Rajpurkar et al., 2016) as well as beat the baseline models in several language inference tasks.

3 Methodology and Data

This section describes the tasks, the dataset, the methodology used and the experiments.

[1] http://ta-cos.org/

[2] https://sites.google.com/site/abusivelanguageworkshop2017/

[3] https://sites.google.com/view/trac1/home

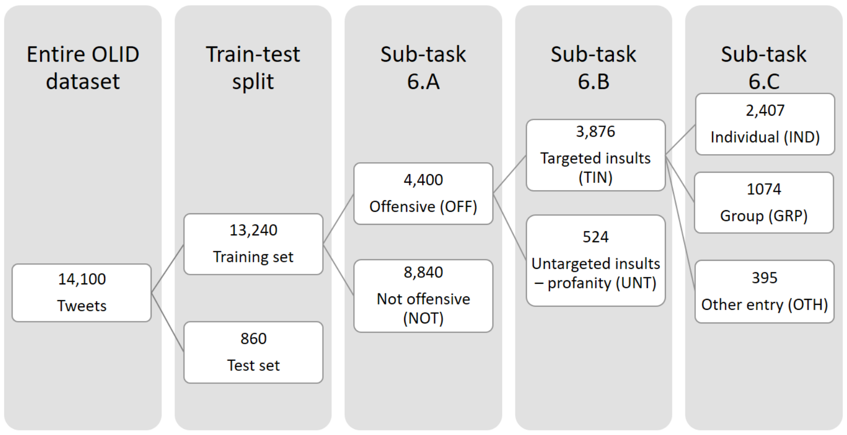


Figure 1: Schema of SemEval-Task 6: OffensEval: Identifying and Categorizing Offensive Language in Social Media. The hierarchy of the sub-tasks and respective dataset sizes.

3.1 Dataset

The SemEval-2019 Shared Task 6: Identifying and Categorizing Offensive Language in Social Media was divided into three sub-tasks, namely offensive language identification (Sub-task A), automatic categorization of offense types (Sub-task B) and offense target identification (Sub-task C). The organizers provided a new dataset called OLID (Zampieri et al., 2019a) which includes tweets labeled according to the three-level hierarchical model. On the very first level, each tweet is labeled as offensive (OFF) or not offensive (NOT). All the offensive tweets are then labeled as targeted insults (TIN) or as untargeted insults (UNT), which simply contain profanity. On the last level, all targeted insults are categorized as targeting an individual (IND), a group (GRP) or other entity (OTH). The dataset contains 14,100 tweets split into training and test sets. The training set containing 13,240 tweets and the test set without labels were made available to the participants for the task. The inspection of the dataset reveals that the classes at first level are slightly imbalanced with the imbalances between classes getting more prominent with each subsequent level. A more detailed breakdown of the dataset is presented in Figure 1. We didn't use any additional datasets in any of the three sub-tasks.

3.2 Methodology

According to the findings from the related work, we decided to test two different types of architectures. First was a pretrained BERT model, which was fine-tuned on the provided dataset for distinguishing offensive and not offensive posts in the Sub-task A. For the sub-tasks B and C, a neural network architecture was developed, which tried to achieve synergy between two types of features that both proved successful in the past approaches to the task at hand, by basing its predictions on a combination of classical bag-of-words features and automatically generated sequence-based features. The three models, as well as their source code, are available for download in a public repository[4].

Three models were trained using the provided dataset, one for each sub-task. In the Sub-task A, the large pretrained BERT transformer with 24 layers of size 1024 and 16 self-attention heads was used for generating predictions on the official test set. A linear sequence classification head responsible for producing final predictions was added on top of the pretrained language model and the whole classification model was fine-tuned on the SemEval input data for 3 epochs. For training, a batch size of 8 and a learning rate of 2e-5 were used. The training dataset for the Sub-task A was randomly split into a training set containing 80% of the tweets and a validation set containing 20% of the tweets. Only a small amount of text preprocessing was needed on the data for the Sub-task A since the dataset already had all Twitter user mentions replaced by @USER tokens and all URLs by URL tokens. Additionally, we lowercased and tokenized the tweets using BERT's built-in tokenizer.

For Sub-task B, the non-offensive tweets were first filtered out of the original dataset. The re-

[4]https://gitlab.com/Andrazp/embeddia-semeval2019

duced dataset had 4400 tweets. To offset the lower quantity of data, we decided to split the dataset into a training set containing 90% of the data and a validation set containing 10% of the data. The second issue with the data was a severe class imbalance as only 12% of tweets in the filtered dataset were labeled as untargeted insults. We decided to resample the dataset in order to minimize the impact of the imbalance on our training. The approach that yielded the best results based on the validation set performance was to randomly remove the instances of the majority class until the classes were balanced. The remaining instances were lowercased and tokenized with the tweet tokenizer from the NLTK package (Bird et al., 2009). Stopwords were also removed from every tweet using an English stopwords list provided in the NLTK package.

As the BERT model was showing worse performance on the resampled data according to the validation set results, a new neural network architecture was devised for this sub-task (Figure 2). The neural architecture takes two inputs. The first input is a term frequency-inverse document frequency (tf-idf) weighted bag-of-words matrix calculated on 1- to 5-grams and character 1- to 7- grams using sublinear term frequency scaling. N-grams with document frequencies less than 5 were removed from the final matrix. Furthermore, the following additional features are generated for each tweet in the training set and added to the tf-idf matrix:

- The number of insults: using a list of English insults,[5] the insults in each tweet are counted and their number is added to the matrix as a feature.

- The length of the longest punctuation sequence: for every punctuation mark that appears in the Python built-in list of punctuations, its longest sequence is found in each tweet. The length of the sequence is then added as a feature.

- Sentiment of the tweets: the sentiment of each tweet is predicted by an SVM model (Mozetič et al., 2016) pretrained on English tweets. The model classifies each tweet as positive, neutral or negative. The predictions are then encoded and added as features.

The second input is word sequences, which are fed into an embedding layer with pretrained 100-dimensional GloVe (Pennington et al., 2014) embedding weights trained on a corpus of English tweets. The pretrained embeddings are additionally fine-tuned during the training process on the dataset for the task. The resulting embeddings are fed to an LSTM layer with 120 units, on the output of which we perform global max pooling. We perform a dropout operation on the max pooling output and the resulting vectors are concatenated with the tf-idf vectors. The resulting concatenation is sent to a fully-connected hidden layer with 150 units, the output of which is fed to a rectified linear unit (RELU) activation function. After performing dropout, final predictions are produced by a fully-connected hidden layer with a sigmoid activation function. For training, we use a batch size of 16 and Adam optimizer with a learning rate of 0.001. We trained the model for a maximum of 10 epochs and validated its performance on the validation set after every epoch. The best performing model was later used for generating predictions on the official test set.

For Sub-task C, the dataset was additionally filtered by removing the tweets that were labeled as non-targeted insults. The class imbalance for this task was even more prominent with only 28% of tweets being labeled as insults targeted towards groups and 10% as targeted insults that do not target an individual or a specific group of people. In light of such class imbalance, the dataset was again undersampled by removing 75% of tweets from the majority class and 50% percent of tweets from the middle class. Due to the dataset being even more aggressively filtered, the 90-10% split from the previous sub-task was kept. A modified version of the neural architecture from Sub-task B was used for prediction. We tried to capture the relationship between insults and their targets using sentence structure information. To this end, we added a third input to the neural architecture that accepts sequences of part-of-speech (POS) tags. First, all the tweets were POS-tagged using the POS tagger from the NLTK package and the resulting POS tag sequences were then fed to a randomly initialized embedding layer. Output embeddings are then fed to an LSTM layer with 120 units, on the output of which we performed global

[5] http://metadataconsulting.blogspot.com/2018/09/Google-Facebook-Office-365-Dark-Souls-Bad-Offensive-Profanity-key-word-List-2648-words.html

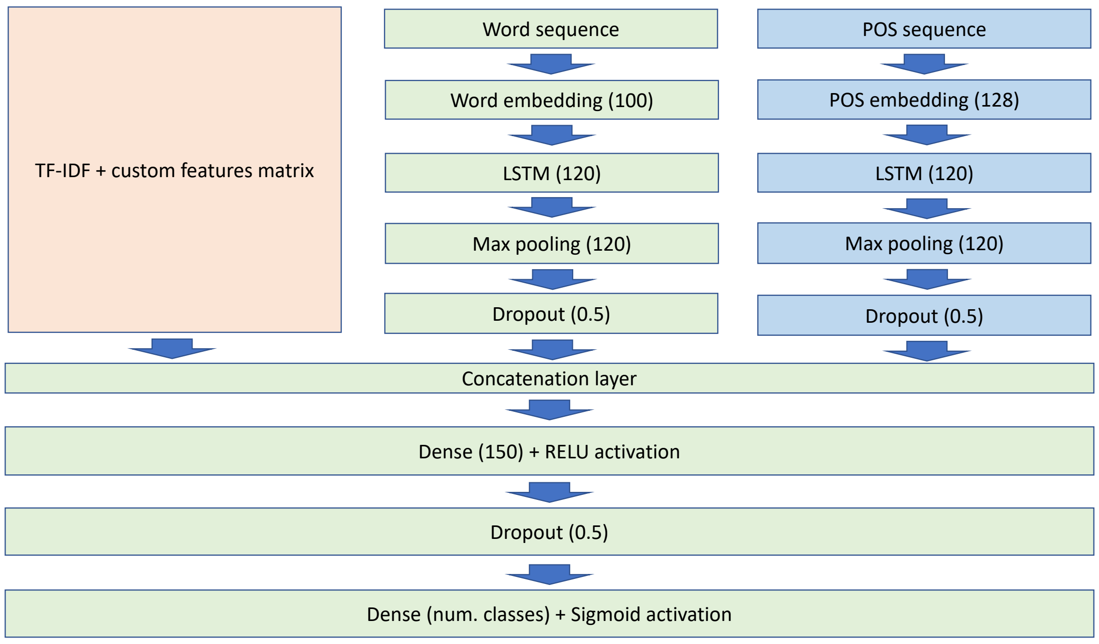


Figure 2: System architecture used in Sub-tasks B and C. Parts of the infrastructure depicted in blue were only used in Sub-task C.

max pooling. Next, dropout was applied, and the resulting vector matrix was then concatenated with the matrices from other inputs and sent to the fully-connected layer (see Figure 2).

4 Results

The results on the official test sets for all three tasks are presented in Table 1. In the Sub-task A, our BERT model, fine-tuned on the provided dataset, achieved a macro-averaged F1 score of 0.808. When we compare this result to other teams participating in the SemEval-2019 OffensEval Sub-task A, we rank fourth.

As the dataset was filtered and the class imbalances became more prominent in the subsequent tasks, the performance of our models started to deteriorate. Even though the undersampling of the dataset to offset class imbalances further reduced the available data, it proved to be the best way to ensure somewhat reliable predictions. The models for Sub-task B and C had macro-averaged F1 scores of 0.663 and 0.613 respectively and placed eighteenth and fifth overall in the SemEval-2019 OffensEval official ranking.

A closer look at the confusion matrices further confirms our claim about the impact of class imbalances on our systems' performance. While the predictions for both classes were fairly accurate in the Sub-task A (Figure 3a), we can see a dwindling performance on the untargeted insults (UNT) class in Sub-task B (Figure 3b) where approximately two thirds of the instances were misclassified as targeted insults (TIN) class on the test set.

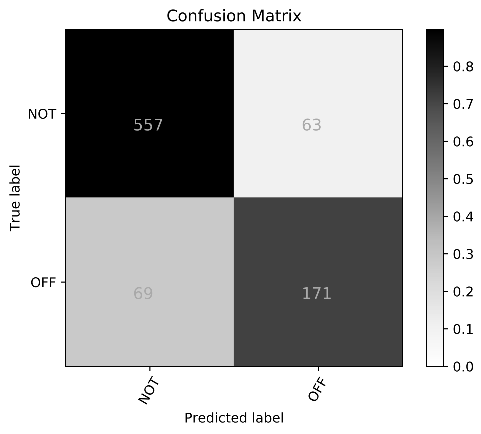
The confusion matrix for Sub-task C (Figure 3c) paints a very similar picture. Even though the majority individual (IND) and middle group (GRP) classes were heavily imbalanced in the original dataset, our model was still able to successfully discriminate between them. However, it again performed subpar on the minority other entity (OTH) class, which was heavily underrepresented compared to the other two. Of the 35 instances in the test set, three out of four were misclassified.

5 Conclusion

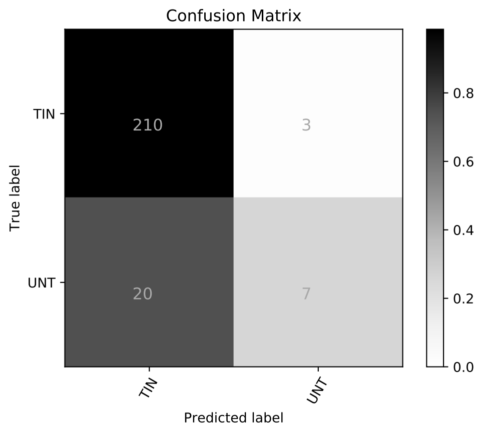
In this paper, we presented the results of the Embeddia team on the SemEval-2019 Task 6: OffensEval: Identifying and Categorizing Offensive Language in Social Media using the dataset provided by the organizers of the task. The task was further divided into three sub-tasks, namely offensive language identification (Sub-task A), automatic categorization of offense types (Sub-task B) and offense target identification (Sub-task C). We trained three models, one for each sub-task. For Sub-task A, we used a BERT model fine-tuned on the OLID dataset, while for the second and third tasks we developed a neural network architecture

Sub-task	System	F1 (macro)	Accuracy
A	BERT	0.8078	0.8465
B	BOW+GloVeLSTM	0.6632	0.9042
C	BOW+GloVeLSTM+POS_LSTM	0.6133	0.7042

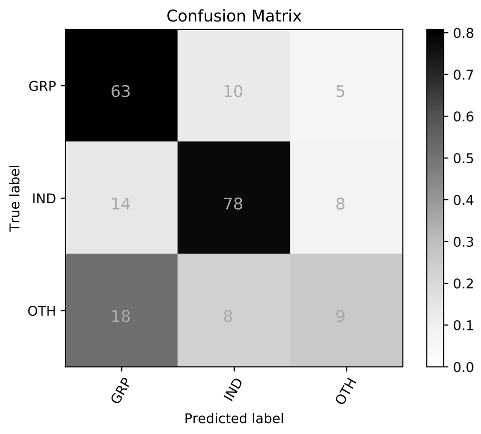
Table 1: Results of the submitted systems for each sub-task.



(a) Confusion matrix for the BERT system, fine-tuned on the provided dataset for Sub-task A.



(b) Confuaion matrix of the two-input neural network with a LSTM based on word sequences and a bag-of-words matrix for Sub-task B.



(c) Confusion matrix of the three-input neural network with an LSTM based on word sequences, LSTM based on part-of-speech tags sequences and a bag-of-words matrix for Sub-task C.

which combines bag-of-words features and automatically generated sequence-based features. Our models ranked fourth, eighteenth and fifth in Subtasks A, B and C, respectively.

We noticed that the class imbalances in the datasets had a significant impact on the performance of our systems and were especially deteriorating for the performance of the BERT system. To counteract the impact of class imbalances we used various techniques to resample the original datasets. While randomly removing instances from the majority classes proved to be the most consistent approach to improve the predictive power of our systems, the effect of the class imbalance persisted.

Our aim for the future is to make the systems more robust to imbalanced data to better generalize over all the classes. Since we already have several models that perform adequately, a good next step would be to implement an ensemble model using a plurality voting or a gradient boosting scheme. We will also conduct an ablation study to identify which features work particularly well for offensive content and hate speech detection.

Acknowledgments

This paper is supported also by the European Unions Horizon 2020 research and innovation programme under Grant No. 825153, EMBEDDIA (Cross-Lingual Embeddings for Less-Represented Languages in European News Media). The results of this publication reflect only the authors views and the Commission is not responsible for any use that may be made of the information it contains. The authors acknowledge also the financial support from the Slovenian Research Agency core research programme Knowledge Technologies (P2-0103). The Titan Xp used for this research was donated by the NVIDIA Corporation.

References

Segun Taofeek Aroyehun and Alexander Gelbukh. 2018. Aggression detection in social media: Using deep neural networks, data augmentation, and pseudo labeling. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 90–97.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Sara Kiesler, Jane Siegel, and Timothy W McGuire. 1984. Social psychological aspects of computer-mediated communication. *American psychologist*, 39(10):1123.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Igor Mozetič, Miha Grčar, and Jasmina Smailović. 2016. Multilingual twitter sentiment classification: The role of human annotators. *PloS one*, 11(5):e0155036.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. Deeper attention to abusive user content moderation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1125–1135.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

LaShel Shaw. 2011. Hate speech in cyberspace: bitterness without boundaries. *Notre Dame JL Ethics & Pub. Pol'y*, 25:279.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Fermi at SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media using Sentence Embeddings

Vijayasaradhi Indurthi[1,3], **Bakhtiyar Syed**[1], **Manish Shrivastava**[1]
Manish Gupta[1,2], **Vasudeva Varma**[1]
[1] IIIT Hyderabad, [2] Microsoft, [3] Teradata
[1]{vijaya.saradhi, syed.b}@research.iiit.ac.in
[1]{m.shrivastava, manish.gupta, vv}@iiit.ac.in
[2]gmanish@microsoft.com
[3]vijayasaradhi.indurthi@teradata.com

Abstract

This paper describes our system (Fermi) for Task 6: OffensEval: Identifying and Categorizing Offensive Language in Social Media of SemEval-2019. We participated in all the three sub-tasks within Task 6. We evaluate multiple sentence embeddings in conjunction with various supervised machine learning algorithms and evaluate the performance of simple yet effective embedding-ML combination algorithms. Our team (**Fermi**)'s model achieved an F1-score of 64.40%, 62.00% and 62.60% for sub-task A, B and C respectively on the official leaderboard. Our model for sub-task C which uses pretrained ELMo embeddings for transforming the input and uses SVM (RBF kernel) for training, scored third position on the official leaderboard.

Through the paper we provide a detailed description of the approach, as well as the results obtained for the task.

1 Introduction

Social media provides anonymity which can be misused to target offensive comments to targeted parties. Users may engage in generating offensive content on social media which may show aggressive behaviour and may also include hate speech. As a result, it is imperative for social media platforms to invest heavily in creating solutions which can identify offensive language and to prevent such behaviour on social media.

Using computational methods to identify offense, aggression and hate speech in user generated content has been gaining attention in the recent years as evidenced in (Waseem et al., 2017; Davidson et al., 2017; Malmasi and Zampieri, 2017; Kumar et al., 2018) and workshops such as Abusive Language Workshop (ALW) [1] and Workshop on Trolling, Aggression and Cyberbullying (TRAC) [2].

[1]https://sites.google.com/view/alw2018

2 Related Work

In this section we briefly describe other work in this area.

Papers published in the last two years include the surveys by (Schmidt and Wiegand, 2017) and (Fortuna and Nunes, 2018), the paper by (Davidson et al., 2017) which presented the Hate Speech Detection dataset used in (Malmasi and Zampieri, 2017) and a few other recent papers such as (ElSherief et al., 2018; Gambäck and Sikdar, 2017; Zhang et al., 2018; Badjatiya et al., 2017).

A proposal of typology of abusive language sub-tasks is presented in (Waseem et al., 2017). For studies on languages other than English see (Su et al., 2017) on Chinese and (Fišer et al., 2017) on Slovene. Finally, for recent discussion on identifying profanity vs. hate speech see (Malmasi and Zampieri, 2018). This work highlighted the challenges of distinguishing between profanity, and threatening language which may not actually contain profane language.

Some of the similar and related previous workshops are Text Analytics for Cybersecurity and Online Safety (TA-COS) [3], Abusive Language Workshop [4], and TRAC [5]. Related shared tasks include GermEval (Wiegand et al., 2018) and TRAC (Kumar et al., 2018).

3 Methodology

3.1 Word Embeddings

Word embeddings have been widely used in modern Natural Language Processing applications as

[2]https://sites.google.com/view/trac1
[3]http://ta-cos.org/
[4]https://sites.google.com/site/alw2018
[5]https://sites.google.com/view/trac1

they provide vector representation of words. They capture the semantic properties of words and the linguistic relationship between them. These word embeddings have improved the performance of many downstream tasks across many domains like text classification, machine comprehension etc. (Camacho-Collados and Pilehvar, 2018). Multiple ways of generating word embeddings exist, such as Neural Probabilistic Language Model (Bengio et al., 2003), Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and more recently ELMo (Peters et al., 2018).

These word embeddings rely on the distributional linguistic hypothesis. They differ in the way they capture the meaning of the words or the way they are trained. Each word embedding captures a different set of semantic attributes which may or may not be captured by other word embeddings. In general, it is difficult to predict the relative performance of these word embeddings on downstream tasks. The choice of which word embeddings should be used for a given downstream task depends on experimentation and evaluation.

3.2 Sentence Embeddings

While word embeddings can produce representations for words which can capture the linguistic properties and the semantics of the words, the idea of representing sentences as vectors is an important and open research problem (Conneau et al., 2017).

Finding a universal representation of a sentence which works with a variety of downstream tasks is the major goal of many sentence embedding techniques. A common approach of obtaining a sentence representation using word embeddings is by the simple and naïve way of using the simple arithmetic mean of all the embeddings of the words present in the sentence. Smooth inverse frequency, which uses weighted averages and modifies it using Singular Value Decomposition (SVD), has been a strong contender as a baseline over traditional averaging technique (Arora et al., 2016). Other sentence embedding techniques include p-means (Rücklé et al., 2018), InferSent (Conneau et al., 2017), SkipThought (Kiros et al., 2015), Universal Encoder (Cer et al., 2018).

We formulate each of the sub-tasks of OffensEval as a text classification task. In this paper, we evaluate various pre-trained sentence embeddings for identifying the offense, hate and aggression. We train multiple models using different machine learning algorithms to evaluate the efficacy of each of the pre-trained sentence embeddings for the downstream sub-tasks as defined in this task. In the following, we discuss various popular sentence embedding methods in brief.

- InferSent (Conneau et al., 2017) is a set of embeddings proposed by Facebook. InferSent embeddings have been trained using the popular language inference corpus. Given two sentences the model is trained to infer whether they are a contradiction, a neutral pairing, or an entailment. The output is an embedding of 4096 dimensions.

- Concatenated Power Mean Word Embedding (Rücklé et al., 2018) generalizes the concept of average word embeddings to power mean word embeddings. The concatenation of different types of power mean word embeddings considerably closes the gap to state-of-the-art methods mono-lingually and substantially outperforms many complex techniques cross-lingually.

- Lexical Vectors (Salle and Villavicencio, 2018) is another word embedding similar to fastText with slightly modified objective. FastText (Bojanowski et al., 2016) is another word embedding model which incorporates character n-grams into the skipgram model of Word2Vec and considers the sub-word information.

- The Universal Sentence Encoder (Cer et al., 2018) encodes text into high dimensional vectors. The model is trained and optimized for greater-than-word length text, such as sentences, phrases or short paragraphs. It is trained on a variety of data sources and a variety of tasks with the aim of dynamically accommodating a wide variety of natural language understanding tasks. The input is variable length English text and the output is a 512 dimensional vector.

- Deep Contextualized Word Representations (ELMo) (Peters et al., 2018) use language models to get the embeddings for individual words. The entire sentence or paragraph is taken into consideration while calculating these embedding representations. ELMo uses

Model	**LR**		**RF**		**SVM-RBF**		**XGB**	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
InferSent	70.32	70.46	70.77	67.26	65.45	60.12	75.52	74.21
Concat-p	69.82	69.95	71.60	70.68	70.37	71.11	75.41	75.23
Lexical Vectors	82.80	82.11	74.42	81.55	79.3	68.3	81.87	81.92
Universal Encoder	74.57	71.07	58.52	74.90	69.67	56.43	75.44	71.37
ELMo	80.00	78.72	73.54	85.20	82.66	73.44	83.27	80.90

Table 1: *Dev* Set Accuracy and Macro-F1 scores (in percentage) for **Sub-Task A**

Model	**LR**		**RF**		**SVM-RBF**		**XGB**	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
InferSent	82.98	80.47	82.29	82.00	80.49	84.02	85.30	83.99
Concat-p	83.17	82.13	80.29	83.64	80.37	82.39	85.17	84.14
Lexical Vectors	76.80	74.16	77.47	81.30	79.3	79.84	79.36	77.63
Universal Encoder	78.57	76.75	58.52	84.90	69.67	56.43	82.41	81.28
ELMo	78.24	76.67	83.54	82.20	82.66	80.72	81.27	79.68

Table 2: *Dev* Set Accuracy and Macro-F1 scores (in percentage) for **Sub-Task B**

Model	**LR**		**RF**		**SVM-RBF**		**XGB**	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
InferSent	66.92	64.98	69.29	65.24	60.49	32.51	68.30	69.03
Concat-p	55.37	60.40	60.29	66.93	66.17	64.35	70.37	68.93
Lexical Vectors	62.80	61.80	64.48	63.44	41.30	29.29	71.87	66.83
Universal Encoder	64.57	60.68	58.52	69.55	61.67	63.47	62.14	69.55
ELMo	80.00	60.48	73.54	64.65	71.66	67.00	69.47	67.76

Table 3: *Dev* Set Accuracy and Macro-F1 scores (in percentage) for **Sub-Task C**

a pre-trained bi-directional LSTM language model. For the input supplied, the ELMo architecture extracts the hidden state of each layer. A weighted sum is computed of the hidden states to obtain an embedding for each sentence.

Using each of the sentence embeddings we have mentioned above, we seek to evaluate how each of them performs when the vector representations are supplied for classification with various off-the-shelf machine learning algorithms. For each of the evaluation tasks, we perform experiments using each of the sentence embeddings mentioned above and show our classification performance on the *dev* set given by the task organizers.

4 Dataset

The data collection methods used to compile the dataset used in OffensEval is described in (Zampieri et al., 2019). Sub-task A (Offensive language Detection) deals with classifying posts as offensive (OFF) vs not (NOT). Sub-task B (Categorization of Offensive Language) deals with categorization of offense as: targeted (TIN) and untargeted (INT). Sub-task C (Offensive Language Target Identification) categorizes the targets of insults and threats as individual (IND), group (GRP), and other (OTH). The overall dataset across the three sub-tasks consists of 14100 posts. Fig. 1 (reproduced from (Zampieri et al., 2019)) shows dataset size details.

A	**B**	**C**	**Training**	**Test**	**Total**
OFF	TIN	IND	2,407	100	2,507
OFF	TIN	OTH	395	35	430
OFF	TIN	GRP	1,074	78	1,152
OFF	UNT	—	524	27	551
NOT	—	—	8,840	620	9,460
All			13,240	860	14,100

Figure 1: Distribution of label combinations in the data (taken from (Zampieri et al., 2019))

		Predicted Label	
True Label		NOT	OFF
	NOT	605	15
	OFF	172	68

Table 4: Sub-task A, ELMo sentence embeddings with SVM classifier using RBF kernel

		Predicted Label	
True Label		TIN	UNT
	TIN	198	15
	UNT	19	8

Table 5: Sub-task B, Concatenated p mean sentence embeddings with XGBoost classifier

5 Results and Analysis

Note that we have not used any external datasets to augment the data for training our models.

In Tables 1, 2, and 3, we provide the dev set macro-averaged F-1 and accuracy for each of the three sub-tasks A, B and C respectively.

We notice the best performance across tasks with ELMo embeddings with SVM (using the RBF kernel).

The confusion matrices for our test set classifications are also given in Tables 4, 5, 6 respectively for each of the sub-tasks A, B and C.

Similar trends are observed for the final classification results on the test set (scored on CodaLab) for the sub-tasks A, B and C in Tables 7, 8, 9 respectively. Our system performed the third best in sub-task C of the 2019 SemEval task.

Overall, this work shows how different set of pre-trained embeddings trained using different state-of-the-art architectures and methods when used with simple machine learning classifiers perform very well for the classification task of categorizing text as offensive or not.

		Predicted Label		
True Label		GRP	IND	OTH
	GRP	52	18	8
	IND	9	85	6
	OTH	11	12	12

Table 6: Sub-task C, Universal Encoder sentence embeddings with XGBoost classifier

System	**F1 (macro)**	**Accuracy**
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
Lexvec	0.4317	0.7233
Concat p-means	0.5572	0.7558
ELMo	**0.6436**	**0.7826**

Table 7: Results for Sub-task A using LexVec, Concatenated p-mean and ELMo sentence embeddings with SVM classifier using RBF kernel

System	**F1 (macro)**	**Accuracy**
All TIN baseline	0.4702	0.8875
All UNT baseline	0.1011	0.1125
Concat p-means	**0.6205**	**0.8583**
InferSent	0.5953	0.8792
Universal	0.5950	0.775

Table 8: Results for Sub-task B. using Concatenated p-mean, InferSent and Universal sentence embeddings with XGBoost classifier

System	**F1 (macro)**	**Accuracy**
All GRP baseline	0.1787	0.3662
All IND baseline	0.2130	0.4695
All OTH baseline	0.0941	0.1643
InferSent	0.4425	0.6009
Universal	**0.6258**	**0.6995**
ELMo	0.5176	0.6103

Table 9: Results for Sub-task C. using InferSent, Universal and ELMo embeddings with XGBoost classifier

6 Conclusions and Future Work

It is also important to note that the experiments are performed using the default parameters, so there is further scope for improvement with a lot of fine-tuning, which we plan on considering for future research purposes. Further, we observe that the class distribution is highly imbalanced due to which there might be a bias introduced by the training algorithms. We plan to explore SMOTE (Chawla et al., 2002) for making the class labels balanced and then train the classification which will prevent the bias towards the unbalanced classes.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *WWW*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Jose Camacho-Collados and Mohammad Taher Pilehvar. 2018. From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research*, 63:743–788.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Darja Fišer, Tomaž Erjavec, and Nikola Ljubešić. 2017. Legal Framework, Dataset and Annotation Schema for Socially Unacceptable On-line Discourse Practices in Slovene. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-Thought Vectors. *arXiv preprint arXiv:1506.06726*.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 467–472.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Andreas Rücklé, Steffen Eger, Maxime Peyrard, and Iryna Gurevych. 2018. Concatenated p-mean word embeddings as universal cross-lingual sentence representations. *arXiv preprint arXiv:1803.01400*.

Alexandre Salle and Aline Villavicencio. 2018. Incorporating subword information into matrix factorization word embeddings. *arXiv preprint arXiv:1805.03710*.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Huei-Po Su, Chen-Jie Huang, Hao-Tsung Chang, and Chuan-Jie Lin. 2017. Rephrasing Profanity in Chinese Text. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

Ghmerti at SemEval-2019 Task 6: A Deep Word- and Character-based Approach to Offensive Language Identification

Ehsan Doostmohammadi[♣, ♠], **Hossein Sameti**[♣], **Ali Saffar**[♠]

[♣]Speech Processing Lab, Department of Computer Engineering,
Sharif University of Technology, Tehran, Iran
[♠]NazarBin, Tehran, Iran
e.doostm72@student.sharif.edu, sameti@sharif.edu,
saffar@nazarbin.com

Abstract

This paper presents the models submitted by Ghmerti team for subtasks A and B of the OffensEval shared task at SemEval 2019. OffensEval addresses the problem of identifying and categorizing offensive language in social media in three subtasks; whether or not a content is offensive (subtask A), whether it is targeted (subtask B) towards an individual, a group, or other entities (subtask C). The proposed approach includes character-level Convolutional Neural Network, word-level Recurrent Neural Network, and some preprocessing. The performance achieved by the proposed model for subtask A is 77.93% macro-averaged F_1-score.

1 Introduction

The massive rise in user-generated web content, alongside with the freedom of speech in social media and anonymity of the users has brought about an increase in online offensive content and anti-social behavior. The consequences of such behavior on genuine users of the social media have become a serious concern for researchers in Natural Language Processing and related fields in recent years.

The shared task number 6 at SemEval 2019, OffensEval (Zampieri et al., 2019b), proposes to model the task of offensive language identification hierarchically, which means identifying the offensive content, whether it is targeted, and if so, the target of the offense. In OffensEval, offensive language is defined as “any form of non-acceptable language (profanity) or a targeted offense, which can be veiled or direct” which includes “insults, threats, and posts containing profane language or swear words” (Zampieri et al., 2019b).

We have participated in the first two subtasks (A and B) of OffensEval with the proposed approach of a deep model consisting of a Recurrent Neural Network (RNN) for word-level and Convolutional Neural Network (CNN) for character-level processing[1]. Character-level processing is beneficial, as offensive comments are likely to follow unorthodox writing styles, contain obfuscated words, or have irregular word separation which leads to tokenization issues (Mehdad and Tetreault, 2016; Nobata et al., 2016). We also experimented with two other methods, a Support Vector Machine (SVM) with TFIDF and count features and another SVM with BERT (Devlin et al., 2018) -encoded sentences as input, both with lower performances comparing with the deep model.

After overviewing the related work in section 2, we discuss the methodology and the data in details in section 3, and the results in section 4. In section 5, we analyze the results and conclude the paper in section 6.

2 Related Work

Offensive language identification has been of interest for researchers in recent years. Early work in the related fields include detection of online trolling (Cambria et al., 2010), racism (Greevy and Smeaton, 2004), and cyberbullying (Dinakar et al., 2012).

Papers published in recent years include (Davidson et al., 2017), which introduces the Hate Speech Detection dataset and experiments with different machine learning models, such as logistic regression, naïve Bayes, random forests, and linear SVMs to investigate hate speech and offensive language, (Malmasi and Zampieri, 2017) which experiments further on the same dataset using SVMs with n-grams and skip-grams features, and (Gambäck and Sikdar, 2017) and (Zhang et al., 2018), both exploring the performance of neural networks and comparing them with other machine

[1]You can find the code of the deep model on this project’s repository on github: github.com/edoost/offenseval

learning approaches. Also, there has been published a couple of surveys covering various work addressing the identification of abusive, toxic, and offensive language, hate speech, etc., and their methodology including (Schmidt and Wiegand, 2017) and (Fortuna and Nunes, 2018).

Additionally, there were several workshops and shared tasks on offensive language identification and related problems, including TA-COS[2], Abusive Language Online[3], and TRAC[4](Kumar et al., 2018), and GermEval (Wiegand et al., 2018), which shows the significance of the problem.

3 Methodology and Data

The methodology used for both subtask A, offensive language identification, and subtask B, automatic categorization of offense types, consists of a preprocessing phase and a deep classification phase. We first introduce the preprocessing phase, then elaborate on the classification phase.

3.1 Preprocessing

The preprocessing phase consists of (1) replacing obfuscated offensive words with their correct form and (2) tweet tokenization using NLTK tweet tokenizer (Bird et al., 2009). In social media, some words are distorted in a way to escape the offense detection systems or to reduce the impertinence. For instance, 'asshole' may be written as 'a$$hole', 'a$sh0le', 'a**hole', etc. Having a list of English offensive words, we can create a list containing most of the possible permutations. Using such a list will ease the job for the classifier and searching in it is computationally cheap. Furthermore, replacing contractions, e.g. 'I'm' with 'I am', and replacing common social media abbreviations, e.g. 'w/' with 'with', were not helpful and were not used to train the final model.

3.2 Deep Classifier

Given a tweet, we want to know if its offensive or not (subtask A), and if the offense is targeted (subtask B). Regarding that both subtasks are problems of binary classification, we used one architecture to tackle both. To define the problem, if we have a tweet x, we want to predict the label y, `OFF` or `NOT` in subtask A, and `TIN` or `UNT` in subtask B. Two representations are therefore created for each input x:

1. x_c which is the indexed representation of the tweet based on its characters padded to the length of the longest word in the corpus. The indices include 256 of the most common characters, plus 0 for padding and 1 for unknown characters.

2. x_w which is the embeddings of the words in the input tweet based on FastText's 600B-token common crawl model (Mikolov et al., 2018).

Then, x_c is fed into an embedding layer with output size of 32 and a CNN layer after that. x_c is then concatenated with x_w and both are fed to a unidirectional RNN with LSTM cell of size 256, the output of which is the input to two consecutive fully-connected layers that map their input to an $\mathbb{R}^{128}$ and an $\mathbb{R}^2$ space, respectively. We also applied dropout of keeping rate 0.5 on CNN's output, x_w, RNN's output, and the first fully-connected layer's output.

The CNN layer consists of four consecutive sub-layers:

1. CNN consisting of 64 filters with kernel size of 2, stride of 1, same padding and RELU activation;

2. max-pooling layer with pool size and stride of 2;

3. another CNN, same as the first one, but with 128 filters;

4. the same max-pooling again.

Finally, we used an AdamOptimizer (Kingma and Ba, 2014) with learning rate of $1\mathrm{e}{-3}$ and batch size of 32 to train the model.

3.3 Baseline Methods

We used two baseline methods for subtask A:

- an SVM with 1- to 3-gram word TFIDF and 1- to 5-gram character count feutrue vectors as input;

- an SVM with BERT representations of the tweets (using average pooling (Xiao, 2018)) as input using `BERT-Large, Uncased` model.

[2] http://ta-cos.org/
[3] https://sites.google.com/site/abusivelanguageworkshop2017/
[4] https://sites.google.com/view/trac1/home

The SVMs were trained for 15 epochs with stochastic gradient descent, hinge loss, alpha of 1e−6, `elasticnet` penalty, and `random_state` of 5. The SVMs were implemented using Scikit-learn (Pedregosa et al., 2011).

3.4 Data

The main dataset used to train the model is Offensive Language Identification Dataset (OLID) Zampieri et al. (2019a). The dataset is annotated hierarchically to identify offensive language (OFFensive or NOT), whether it is targeted (Targeted INsult or UNTargeted), and if so, its target (INDividual, GRouP, or OTHer). We divided the 13,240 samples in the training set into 12,000 samples for training and 1,240 samples for validation.

As neural networks require huge amount of training data, we tried adding more data from the dataset of the First Workshop on Trolling, Aggression, and Cyberbullying (TRAC-1) (Kumar et al., 2018) which was not helpful. However, adding the training data from Toxic Comment Classification Challenge on Kaggle (Conversation AI, 2017) increased the macro-averaged F_1-score on the validation set by $\sim 2\%$. This data comprises tweets with positive and negative tags in six categories: `toxic`, `severe_toxic`, `obscene`, `threat`, `insult`, `identity_hate`. We only used `toxic` and `severe_toxic` positive samples as `OFF` and the ones with no positive label in any category as `NOT`. None of the data from other categories, either positive or negative, were included in the additional training data. After that, we were left with 109,236 samples, most of which were labeled as `NOT`. To balance `OFF` and `NOT` samples, 84,626 of `NOT` samples were randomly removed. In the end, 12,305 `OFF` and 12,305 `NOT` samples were added to the training data.

4 Results

Finally, we trained the baseline models in 3.3 and the model described in 3.2 using the combination of the OLID training data and the data from Toxic Comment Classification Challenge (which is described in 3.4).

You can see the macro-averaged F_1-score and accuracy on the test set for the baseline scores provided by task organizers, baseline methods we used (on both training and validation data), and the deep classifier model (DeepModel) in table 1. DeepModel is trained on the training data (not including the validation data) and DeepModel+val on the combination of the training and validation data. The best performance is in bold.

System	Macro F_1	Accuracy
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
SVM	0.7452	0.8011
BERT-SVM	0.7507	0.8011
DeepModel	0.7788	0.8326
DeepModel+val	**0.7793**	**0.8337**

Table 1: Results for subtask A

The best performance belongs to DeepModel+val by a margin of more than 2.8 percent, with the best baseline performance, BERT-SVM. However, it should be mentioned that the results in the first two rows belong to a model trained only on OLID. You can see the confusion matrix for the best performance in figure 1.

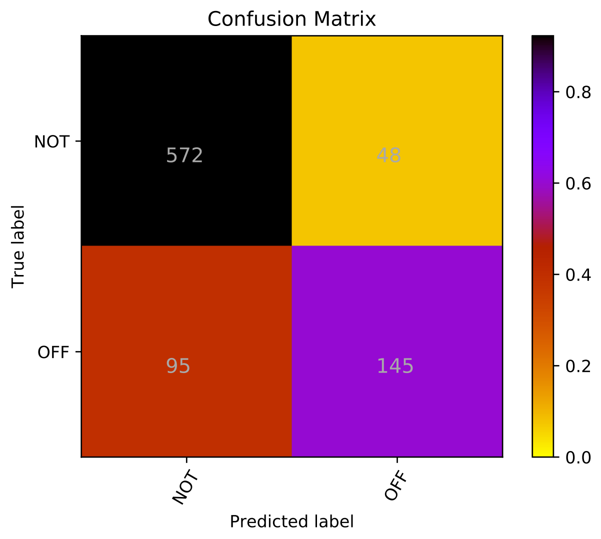


Figure 1: The confusion matrix for DeepModel+val in subtask A

From the confusion matrix we can see that the performance of DeepModel+val on `NOT` is quite good, but not on `OFF`. You can see the detailed results of DeepModel+val in table 2.

	Precision	Recall	F_1-score
`NOT`	0.8576	0.9226	0.8889
`OFF`	0.7513	0.6042	0.6697

Table 2: Detailed DeepModel+val results in subtask A

In subtask B, DeepModel+val outperformed the baseline results by a large margin, like subtask A. The results for subtask B are presented in table 3.

System	Macro F_1	Accuracy
All TIN baseline	0.4702	0.8875
All UNT baseline	0.1011	0.1125
DeepModel	0.6065	0.8583
DeepModel+val	**0.6400**	**0.8875**

Table 3: Results for subtask B

This time, adding the validation data made a considerable difference, as the training data for subtask B is fewer. You can see the confusion matrix for DeepModel+val in figure 2.

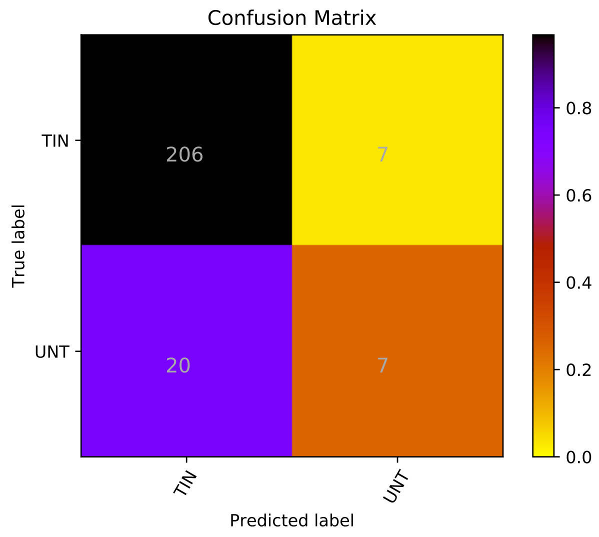


Figure 2: The confusion matrix for the DeepModel+val in subtask B

The confusion matrix shows that the performance of the model is good for `TIN`, but poor for `UNT`. Table 4 shows the detailed results for DeepModel+val in subtask B, which indicates that the imbalance is worse than subtask A and the poor performance on `UNT` is mainly due to low recall.

	Precision	Recall	F_1-score
`TIN`	0.9115	0.9671	0.9385
`UNT`	0.5000	0.2593	0.3415

Table 4: Detailed DeepModel+val results in subtask B

5 Analysis

In subtask A, DeepModel+val outperformed the second best method, BERT-SVM, by 2.86% Macro F_1-score. BERT-SVM results, however, were not much better than the SVM with TFIDF and count features, probably due the fact that the BERT model requires fine-tuning for more task-specific representations.

The majority of DeepModel+val's errors are in `OFF` class and can be categorized into (1) sarcasm: the model is unable to detect sarcastic language which is even difficult for humans to detect; (2) emotion: discerning emotions, such as anger, seems to be a challenge for the model; (3) ethnic and racial slurs, etc. Solving these problems require a more comprehensive knowledge of the context and the language, which was examined in works such as (Poria et al., 2016) and improved the results. However, experimenting with emotion embeddings in the current work was not helpful and did not appear in the final results. Being aware of the emotion of the text, personality of the author, and sentiment of the sentences is helpful to detect offensive language, as many offensive contents have an angry tone (ElSherief et al., 2018) or do not contain profane language (Malmasi and Zampieri, 2018). One can also make use of the benefits of BERT's context and sentence sequence awareness by fine-tuning it on the training data, which is computationally expensive and was not feasible for the authors of this paper.

6 Conclusion

In this paper, we introduced Ghmerti team's approach to the problems of 'offensive language identification' and 'automatic categorization of offense type' in shared task 6 of SemEval 2019, OffensEval. In subtask A, the neural network-based model outperformed the other methods, including an SVM with word TFIDF and character count features and another SVM with BERT-encoded tweets as input. Furthermore, analysis of the results indicates that sarcastic language, inability to discern the emotions such as anger, and ethnic and racial slurs constitute a considerable portion of the errors. Such deficiencies demand larger training corpora and variety of other features, such as information on sarcasm, emotion, personality, etc.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Erik Cambria, Praphul Chandra, Avinash Sharma, and Amir Hussain. 2010. Do not feel the trolls. *ISWC, Shanghai*.

Conversation AI. 2017. Toxic comment classification challenge: Identify and classify toxic online comments.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Karthik Dinakar, Birago Jones, Catherine Havasi, Henry Lieberman, and Rosalind Picard. 2012. Common sense reasoning for detection, prevention, and mitigation of cyberbullying. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2(3):18.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Edel Greevy and Alan F Smeaton. 2004. Classifying racist texts using a support vector machine. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 468–469. ACM.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 467–472.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Yashar Mehdad and Joel Tetreault. 2016. Do characters abuse more than words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive Language Detection in Online User Content. In *Proceedings of the 25th International Conference on World Wide Web*, pages 145–153. International World Wide Web Conferences Steering Committee.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, and Prateek Vij. 2016. A deeper look into sarcastic tweets using deep convolutional neural networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1601–1612.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval*.

Han Xiao. 2018. bert-as-service. `https://github.com/hanxiao/bert-as-service`.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

HAD-Tübingen at SemEval-2019 Task 6: Deep Learning Analysis of Offensive Language on Twitter: Identification and Categorization

Himanshu Bansal[1]
University of Tübingen

Daniel Nagel[2]
University of Tübingen

Anita Soloveva[3]
University of Tübingen,
Lomonosov MSU

himanshu.bansal[1], daniel.nagel[2], anita.soloveva[3]@student.uni-tuebingen.de

Abstract

This paper describes the submissions of our team, HAD-Tübingen, for the SemEval 2019 - Task 6: "OffensEval: Identifying and Categorizing Offensive Language in Social Media". We participated in all the three sub-tasks: Sub-task A - "Offensive language identification", sub-task B - "Automatic categorization of offense types" and sub-task C - "Offense target identification". As a baseline model we used a Long short-term memory recurrent neural network (LSTM) to identify and categorize offensive tweets. For all the tasks we experimented with external databases in a postprocessing step to enhance the results made by our model. The best macro-average F_1 scores obtained for the sub-tasks A, B and C are 0.73, 0.52, and 0.37, respectively.

1 Introduction

The use of offensive language is an ubiquitous problem one faces when using social networking services like Twitter. Users of such services often take advantage of the anonymity of the individual platforms for using the computer-mediated communication to engage in offensive behaviour against individuals, groups and/or organizations. Due to increasing problems with offensive language and a raising demand for offensive language detection on platforms like Twitter, tasks, similar to the current one have already become popular for several different languages: English (Waseem et al., 2017), German (Wiegand et al., 2018) and Spanish (Rosso et al., 2018). With increasing popularity of Twitter, over 1.48 billion users (June 2013) and still new accounts signing up every day, the need for improvement on tackling the well known problem of insults inside the platform has become more and more necessary.

The Twitter platform[1] describes itself as a connection to "what's happening in the world and what people are talking about right now". For this reason alone, its data attracts more and more NLP researchers all over the world. "Tweets", the messages one can send over this platform can be described as micro-texts, limited to 280 characters, over which users can interact with each other or simply post statements. Since the input is up to the user, one could include misspellings, emoticons, hashtags but also slang and abusive words, what makes those messages a valuable source for different analyses.

As was mentioned in the beginning, the goal of this paper is to consider our approach for the SemEval 2019 - Task 6: "OffensEval: Identifying and Categorizing Offensive Language in Social Media", for task information (see Zampieri et al. 2019b) and for dataset description (see Zampieri et al. 2019a). We took part in all of the three subtasks, using an LSTM based classifier. In the remainder of the paper, we describe our methods and discuss both our results and suggestions for further work.

2 System description

Neural network models have recently gained more and more popularity for text classification tasks, since they perform quite efficiently in modeling of sequences and offer advantages for computation. For this competition, we used unidirectional LSTM, where the recurrent component took a sequence of words as an input. We set the basic parameters in the model as follows: 30 as the number of epochs, a batch size of 43 for sub-task A, since it was the smallest batch size that the 860 tweets could be divided by, where our model

[1]https://twitter.com/

still performed well. For the other sub-tasks we went with 30 and 71 as batch sizes for the test sets of 240 and 213 tweets, accordingly. We used 4 hidden layers with 50 neurons per each, since our overall score declined by decreasing and increasing their number. Our dropout ratio was set to 0.95, the embedding size to 100, learning rates varied between submissions from 0.003 to 0.005.

The model was implemented in Python and makes use of Tensorflow (Abadi et al., 2015) and Scikit-learn (Pedregosa et al., 2011) libraries for training the classifier. We optimized our architecture parameters by predictions of support vector machine (SVM) model, described in (Rama and Çöltekin, 2017) and (Çöltekin and Rama, 2018). It used 'bag of n-grams' as features, and took not only word n-grams, as in our LSTM based model, but combined character and word n-grams, weighted by sublinear TF-IDF scaling. We picked the epoch with the best F_1-score for each parameter setting according to these SVM predictions. Our repository can be found on github https://github.com/cicl2018/semeval-2019-task-6-HAD.

2.1 Preprocessing

For neural network classification, data preprocessing has a great impact on the system's performance. Thus, at least one step from the following procedure was applied for all the submissions:

- lowercasing, since uppercased words can be both offensive and not
- hashtag parsing (e.g. #retrogaming → # retro gaming) (see, Baziotis et al. 2017)
  This tool is trained on 2 big corpora:
  - English Wikipedia
  - a collection of 330 million English Twitter messages
- removing tokens, containing "@USER"
  The user names are not given, thus this information is irrelevant for the classification task.
- character normalization
  We removed all the following charachters " : . , — ˜ ", digits and single quotation marks except for abbreviations and possessors (e.g. *u're* → *u're*, but *about'* → *about*)
- using '=', '!', '?' and '/' as token splitters (e.g. *something!important* → *something important*)

2.2 Sub-task A - Offensive language identification

Sub-task A was a binary classification task. The goal was to identify whether the post is offensive (OFF) or not (NOT). The provided tweets were labeled as OFF if they contained any form of non-acceptable language or a targeted offense, and labeled as NOT in any other case.

2.2.1 System pipeline for sub-task A

Figure 1 describes the system architecture for sub-task A. For each of the three submissions we tried different approaches.

1. All the preprocessing steps (Section 2.1) + LSTM classifier with the use of SVM predictions, (see Section 2.2.2 and green arrows in Figure 1).
2. All the preprocessing steps + LSTM classifier with SVM predictions + additional manually created offensive word list, (see Section 2.2.3 and black arrows in Figure 1).
3. Hashtag parsing as a single preprocessing step + LSTM classifier with SVM predictions, (see Section 2.2.4 and red arrows in Figure 1).

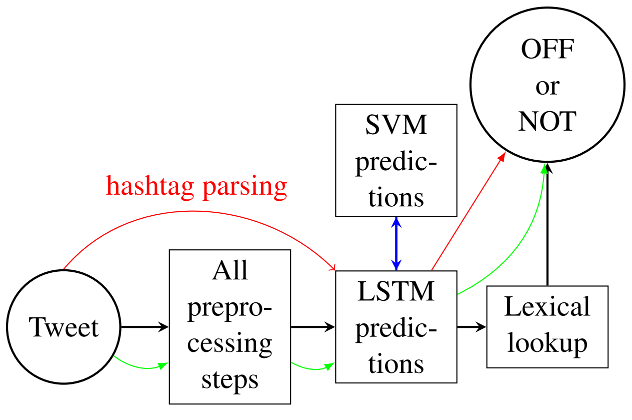


Figure 1: Pipeline of sub-task A

2.2.2 Submission 1, Sub-task A

In our first submission, we fed the preprocessed data into our LSTM model, setting the configurations (e.g. a learning rate of 0.003), according to the outcome of SVM predictions (Figure 1: green arrows).

2.2.3 Submission 2, Sub-task A

For the second submission we used a manually created additional offensive word list. After all the preprocessing steps, we ran the model with the same configurations as in the first submission except for the learning rate of 0.005, picking the epoch with the best F_1-score regarding SVM predictions. Then we postprocessed the results by using external manually collected offensive vocabulary, reannotating the tweets as offensive, if they contained abusive words from this list, but were labeled as not offensive by our model (Figure 1: black arrows).

2.2.4 Submission 3, Sub-task A

As a third submission, we preprocessed raw tweets only by hashtag parsing and let an LSTM model with a learning rate of 0.005 classify the data, choosing the epoch with the best F_1-score, according to the SVM predictions (Figure 1: red arrows).

2.3 Sub-task B - Automatic categorization of offense types

Sub-task B was a classification task of targeted (TIN) vs. untargeted (UNT) tweets. The test set contained only offensive (OFF) posts from the first sub-task. Tweets were considered as targeted, if they were insults/ threats to an individual or group, untargeted in any other case. For this sub-task we reduced the initial training set of 13.240 tweets to 4300, removing the tweets labelled with NOT, since non-offensive tweets would not add any improvement to the learning model and might even distort the learning process.

2.3.1 System pipeline for sub-task B

The system architecture for this sub-task is illustrated in Figure 2. Since the number of the representative tweets in the training data differed between categories a lot (i.e. 524 and 3876 for UNT and TIN, respectively), we used a weighted cross entropy to balance the data. Like in sub-task A, our approaches varied between submissions but this time we handed in 2 submissions.

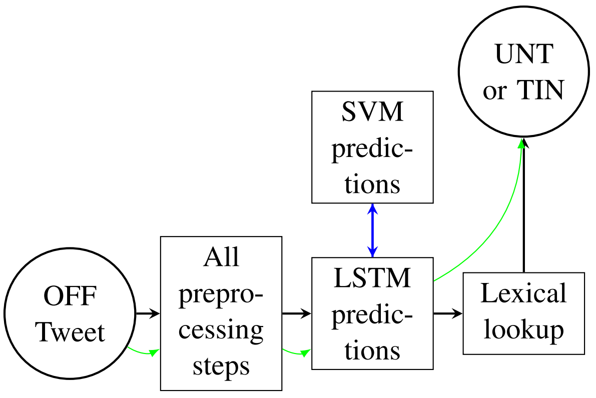


Figure 2: Pipeline of sub-task B

2.3.2 Submission 1, Sub-task B

The architecture of the first submission in this sub-task is very much similar to the first submission in sub-task A with the only difference being that a learning rate was changed to 0.005 (Figure 2: green arrows).

2.3.3 Submission 2, Sub-task B

For the second submission we added a postprocessing step, where we reannotated the tweets that comprised particular word forms from a manually created list of potential insult victims as targets (Figure 2: black arrows). This database included following four parts:

- Names of representatives of top twitter profiles from the USA, the UK, Saudi Arabia, Brazil, India and Spain, since these countries have the most Twitter users[2] and Iran, Iraq, Turkey, Russia and Germany, because we predicted a possible aggression towards the users from these countries. The data was obtained from `https://www.socialbakers.com/statistics/twitter/profiles/`.
- A list of ethnic slurs, mostly extracted from `https://en.wikipedia.org/wiki/List_of_ethnic_slurs`
- A list of name-callings, primarily collected from `https://www.urbandictionary.com/`
- A list of 2nd and 3rd personal pronouns and abbreviations with them (e.g. *you*, *they've* etc.)

[2]This statistic can be found on `https://www.statista.com/statistics/242606/number-of-active-twitter-users-in-sele\protect\discretionary{\char\hyphenchar\font}{}{}cted-countries/`

2.4 Sub-task C - Offense target identification

The third sub-task addressed offense target identification. This time we had three categories to choose from: Individual (IND), group (GRP), or other (OTH). The tweets were labeled as individually targeted, if a potential victim was a famous person, a named IND or an unnamed person interacting in the conversation. It was labeled as GRP, if the tweet was offensive with respect to a group of people considered as a unity due to the same ethnicity, gender or sexual orientation, political affiliation, religious belief, or similar, and was labelled as OTH, if the tweet intended to abuse an organization, a situation, an event, or an issue. The test data contained 213 offensive targeted tweets from sub-task B. The training set of 4300 offensive tweets was reduced to 3909 targeted ones for this sub-task.

2.4.1 System pipeline for sub-task C

The system architecture for this sub-task is illustrated in Figure 3.

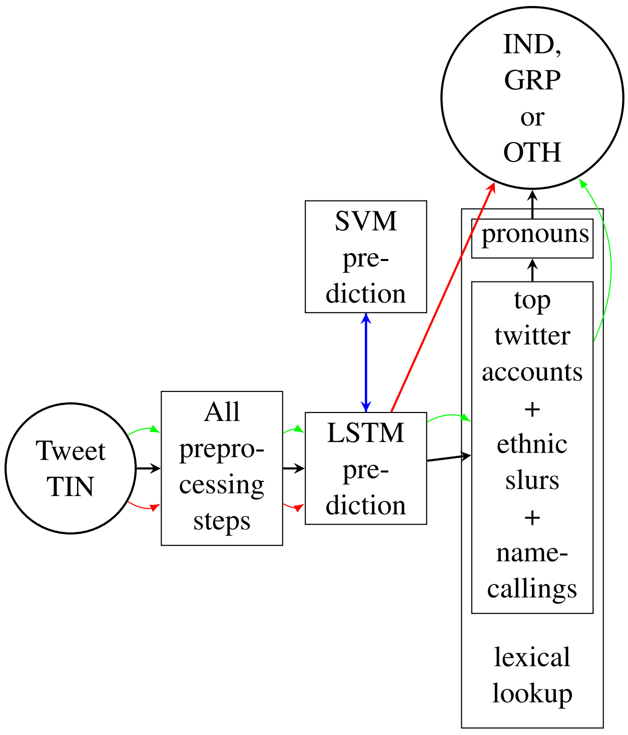


Figure 3: Pipeline of sub-task C

2.4.2 Submission 1, Sub-task C

This submission is reminiscent of the two previous first submissions, but the batch size was set to 71 and the learning rate to 0.003 (Figure 3: red arrows).

2.4.3 Submission 2, Sub-task C

In the second submission, we postprocessed the classified data, using the following datasets:

- Names of representatives of top twitter profiles from the USA, the UK, Saudi Arabia, Brazil, India, Spain, Iran, Iraq, Turkey, Russia and Germany. The data was obtained from `https://www.socialbakers.com/statistics/twitter/profiles/`:
  - celebrities and society/politics industries for identifying individual targets
  - community/political and community/religion industries for recognizing group targets
  - places, brands and entertainment/event industry for other targets
- A list of ethnic slurs, (see Section 2.3.3), for identifying group targets
- A dataset of name-callings, (see Section 2.3.3), for recognizing individual victims

These datasets helped to classify the categories IND, GRP or OTH by looking them up in our lists. (Figure 3: green arrow).

2.4.4 Submission 3, Sub-task C

The third submission differed from the previous one only in adding a list of 2nd and 3rd personal pronouns including their contractions to the existing database for the postprocessing step. We decided to try an approach with personal pronouns despite the fact, that they can both target individuals (e.g. "Take it out, you fucking wanker, or I'll take you out".) and groups (e.g. "All you democrats suck, and your momma's fat!").

3 Results

The results presented below were obtained using the macro-averaged F_1-score, provided by the organisers of OffensEval 2019. They included accuracy as well for comparison. Random baseline generated results by assigning the same labels for all instances were also added to the result Table 1. For example, "All OFF" in sub-task A represented the performance of a system that labels everything as offensive.

System	F1 (macro)	Accuracy
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
LSTM + Prepr.	0.6652	0.7337
LSTM + Prepr. + Lex. lookup	0.6487	0.7349
LSTM + Hashtag parsing	**0.7327**	**0.7977**

Table 1: Results for Sub-task A.

The best results for the first sub-task were produced by the simplest approach, which included only hashtag parsing as a preprocessing step and an LSTM based classifier with configurations, set according to SVM predictions. A plausible explanation to the bad performance of the second submission with a lexical lookup is that a task-specific lexicon should better be used as an input feature, which can only influence data classification, rather than as a decisive postprocessing step.

For sub-task B one can see the scores of our two submissions in Table 2. As before, the organizers have also included random baseline generated results by assigning the same labels for all instances.

System (Sumbission)	F1 (macro)	Accuracy
All TIN baseline	0.4702	0.8875
All UNT baseline	0.1011	0.1125
LSTM + Prepr.	**0.5246**	**0.8417**
LSTM + Prepr. + Lex. lookup	0.5022	0.8833

Table 2: Results for Sub-task B.

The best results for this sub-task were also achieved only by applying preprocessing steps to an LSTM model. Most likely, the problem was that our external dataset largely aimed to recognize names of top twitter accounts, which most frequently occur as usernames in tweets. However, in our case they were anonymized in both training and test sets (@USER). Last table shows the scores of our submissions for sub-task C:

System (Submission)	F1 (macro)	Accuracy
All GRP baseline	0.1787	0.3662
All IND baseline	0.2130	0.4695
All OTH baseline	0.0941	0.1643
LSTM + Prepr.	0.2027	0.3099
LSTM + Prepr. + Lex. lookup without Pronouns	0.3582	0.3709
LSTM + Prepr. + Lex. lookup with Pronouns	**0.3769**	**0.4883**

Table 3: Results for Sub-task C.

For the last sub-task, which was devoted to categorizing targets of offense, a considerable increase in F_1-score can be observed by using the external datasets for postprocessing. Hence, the results showed that using a lexical lookup could be much more efficient in categorizing the possible victims than in identifying the presence of aggression per se. Below one can also find the confusion matrices of our best runs:

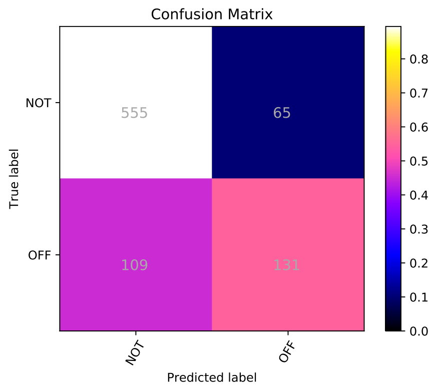


Figure 4: Sub-task A, HAD-Tübingen LSTM + Hashtag parsing.

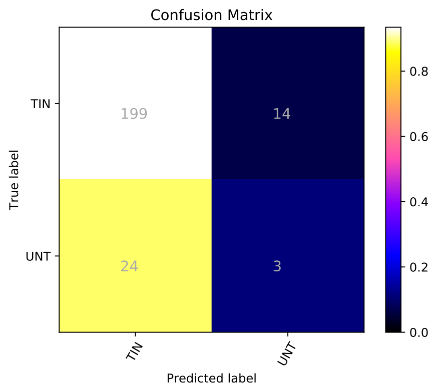


Figure 5: Sub-task B, HAD-Tübingen LSTM + Preprocessing.

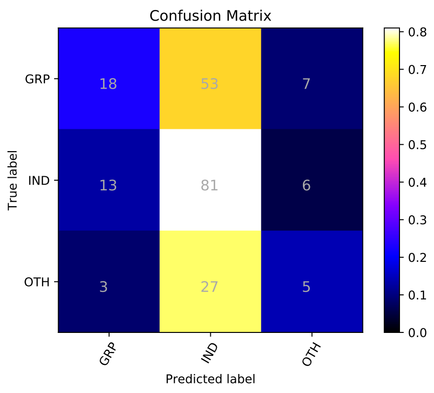


Figure 6: Sub-task C, HAD-Tübingen LSTM + Preprocessing + Lexical lookup with Pronouns.

It is also worth mentioning that a model choice and its settings should be made according to the training set size. In our case, the volume differed significantly for all the sub-tasks. However, a significantly lower performance of all the submissions can be observed on the last sub-task with the smallest training set.

4 Conclusion and future work

In our paper we presented the contribution of HAD-Tübingen to the OffensEval 2019 (SemEval 2019 - Task 6). Our approach combines sentence simplification as a preprocessing step and a lexical lookup as a postprocessing step with an unidirectional LSTM with 4 hidden layers. We picked the epochs according to the best F_1-score for our model configurations, according to SVM predictions. We found out that simple LSTM models are not likely to outperform SVM in such classification tasks. However, as a next possible step in working with an LSTM based classifier, could be using an external task-specific lexicon as an input feature to our model, but not as a postprocessing step. We would also like to make use of the pretrained vectors from Fastext library that are based on sub-word character n-grams for improving our model.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. Tensorflow: Large-scale machine learning on heterogeneous distributed systems.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at SemEval-2017 task 4: Deep LSTM with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada. Association for Computational Linguistics.

Çağrı Çöltekin and Taraka Rama. 2018. Tübingen-oslo at Semeval-2018 task 2: Svms perform better than rnns in emoji prediction. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 34–38. Association for Computational Linguistics.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Taraka Rama and Çağrı Çöltekin. 2017. Fewer features perform well at native language identification task. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 255–260. Association for Computational Linguistics.

Paolo Rosso, Julio Gonzalo, Raquel Martínez, Soto Montalvo, and Jorge Carrillo de Albornoz, editors. 2018. *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages*. Sevilla, Spain.

Zeerak Waseem, Wendy Hui Kyong Chun, Dirk Hovy, and Joel Tetreault, editors. 2017. *The First Workshop on Abusive Language Online: Proceedings of the Workshop*. Association for Computational Linguistics (ACL), Vancouver, Canada.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of the GermEval 2018 Workshop*, pages 1–10, Vienna, Austria. Austrian Academy of Sciences.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

HHU at SemEval-2019 Task 6: Context Does Matter - Tackling Offensive Language Identification and Categorization with ELMo

Alexander Oberstrass[1] **Julia Romberg**[1] **Anke Stoll**[2] **Sefan Conrad**[1]

[1]Institute of Computer Science, Heinrich Heine University Düsseldorf, Germany
alexander.oberstrass@hhu.de
{romberg,conrad}@cs.uni-duesseldorf.de

[2]Department of Social Sciences, Heinrich Heine University Düsseldorf, Germany
anke.stoll@hhu.de

Abstract

We present our results for OffensEval: Identifying and Categorizing Offensive Language in Social Media (SemEval 2019 - Task 6). Our results show that context embeddings are important features for the three different subtasks in connection with classical machine and with deep learning. Our best model reached place 3 of 75 in sub-task B with a macro F_1 of 0.719. Our approaches for sub-task A and C perform less well but could also deliver promising results.

1 Introduction

User generated content in social media platforms such as Twitter often includes high levels of rude, offensive, or sometimes even hateful language. The increasing vulgarity in online discussions and user comment sections have recently been discussed as relevant issues in society as well as in science.

The identification of offensiveness, aggression, and hate speech in user-generated content has been addressed in recent research (Waseem et al., 2017; Davidson et al., 2017a; Malmasi and Zampieri, 2018) and previous shared tasks (Wiegand et al., 2018; Kumar et al., 2018). However, detecting such content automatically is still challenging. We developed classification models to identify *offensive language*, different *categories of offense types*, and *targets of offensive language* throughout the SemEval-2019 challenge on Identifying and Categorizing Offensive Language in Social Media (Zampieri et al., 2019b).

2 Methodology and Data

In this section, we introduce the datasets, the features and classifiers we use.

2.1 Datasets

The training dataset provided for this task is further described in Zampieri et al. (2019a). For sub-task A: Offensive Language Detection 4.400 offensive (*OFF*) and 8.840 not offensive (*NOT*) tweets are given for the training and 240 offensive plus 620 not offensive tweets are given as test data. Sub-task B: Categorization of Offensive Language is provided with a training set of 3876 targeted insult (*TIN*) and 524 untargeted (*UNT*) tweets and a test set of 213 targeted insult plus 27 untargeted tweets. The train data distribution for sub-task C: Offensive Language Target Identification are 2407 tweets targeting an individual (*IND*), 1074 targeting a group (*GRP*) and 395 targeting any other category (*OTH*). The test data given counts 100, 78 and 35, respectively.

We use Davidson et al. (2017b)'s dataset for hate speech detection as an additional source of knowledge and to address the problem of overfitting in our deep learning models. Tweets were collected and filtered using a lexicon of common hate speech terms. The remaining tweets were then each classified by at least 3 CrowdFlower users into the three categories *hate speech*, *offensive language* or *none*. Each tweet was assigned a label, which was chosen based on a majority vote. In order to match the labels to this task, we merged the categories: *hate speech* and *offensive language* into a single offensive label *OFF* and renamed the label *none* to *NOT*. This resulted in 20620 *OFF* and 4163 *NOT* labeled additional tweets. It should be noted that our assumption to equate hate speech and offensive language does not apply in general.

2.2 Features

Word and Char-Level Features We transformed the tweet texts into tf-idf weighted bag of words and bag of chars representations using scikit-learn

(Pedregosa et al., 2011). The feature *CHARS* is built from bigrams to 6-grams on character level. For the feature *WORDS* we use unigrams and bigrams on word level, excluding stop words. In order to reduce variance in the text data, we also built the feature *STEMS* with unigrams and bigrams. For this, we applied the TweetTokenizer from NLTK (Bird et al., 2009) that removes user name handles from a tweet and replaces repeated characters with a sequence of only 3. Afterwards, the word tokens are reduced to their stems using the Snowball Stemming algorithm (Porter et al., 2002).

Named Entities Named entities in a tweet can indicate whether this tweet is person-related or relates to e.g. religious or political groups or organizations. This information might help with the target identification. The feature *NE* is implemented using the named entity recognition of spacy[1], building a bag of named entities for every tweet preserving all OntoNotes5 (Weischedel et al., 2013) named enity types, which results in an 18-dimensional feature vector per tweet.

Grammatical Number of Nouns and Pronouns The grammatical number of a noun might also indicate if a tweet insults a specifiable target. We use spacy's part-of-speech tagger, which uses the OntoNotes5 tagging scheme. The feature *ND* models the noun distribution of singular and plural nouns and is implemented building a bag of tags for the tags NNS, NN, NNP and NNPS, normalized by the tweets' token count. Contrarily, feature *SPNR* describes the ratio of singular noun tags (NN and NNP) and plural noun tags (NNPS and NNS) per tweet. The first value is divided by the latter one, smoothing both by adding 1. Another feature to separate single person targets from groups is the absolute count of single pronoun words (he, she, him, her, his, hers, himself, herself) per tweet, hereinafter referred to as *SPC*. It should be noted that the corresponding procedure for plural pronouns could not bring any benefit in our evaluations and is therefore not further described.

Dependencies Another consideration was whether syntactic dependency relations could provide information for one of the three sub-tasks. For the feature *DEP*, every tweet is transformed into the corresponding list of dependency labels. These representations are subsequently used to build n-grams in the range (2,4) which are then vectorized using tf-idf.

Global Vectors for Word Representation (GloVe) Feeding words into machine learning models often requires a meaningful vector representation that captures syntactic and semantic information. We use GloVe word embeddings (Pennington et al., 2014), an unsupervised learning algorithm for obtaining vector representations of words. A set of pre-trained word vectors, trained on over 2 billion tweets, containing vector representations for over 1.2 million words in various dimensions (25, 50, 100, 200) is publicly available on the author's website[2]. In this work, we use 200 dimensions.

Embeddings from Language Models (ELMo) Traditional pre-trained word representations merely contain meaning based on statistical information and therefore struggle with word-sense disambiguation. Since offensive words change their meaning greatly depending on their context, a contextualizing method would help to tackle this task. ELMo (Peters et al., 2018) is a novel approach on creating word representations and shows a significant improvement of state of the art systems on many benchmarks. It models a function using character-based word representations and bidirectional long-short term memories of not only each single word, but also the entire input sentence. Thus, it can be useful for solving the problem of processing ambiguous words that are not offensive by themselves but could point to offensive language depending on the context they are used in. For these reasons, we expect ELMo to increase the results of this task. The pre-trained model can be downloaded from TensorFlow Hub[3].

χ^2 Feature Selection To quantify the contribution of a feature, we choose the χ^2 test statistic, that excludes features that are most likely to be independent from a class, and keep the k features with the highest values for χ^2.

2.3 Classifier

We use logistic regression (Nelder and Wedderburn, 1972), support vector machines (Cortes and Vapnik, 1995) and neural networks with long short-term memory units (Hochreiter and Schmidhuber, 1997) for the different classification prob-

[1] `https://spacy.io/`

[2] `https://nlp.stanford.edu/projects/glove/`

[3] `https://tfhub.dev/google/elmo/2`

lems. The classifiers are implemented with scikit-learn and TensorFlow (Abadi et al., 2015).

Logistic regression has been used by several teams of the GermEval shared task on offensive language identification (Wiegand et al., 2018) with promising results. We therefore decided to use it as a baseline approach. In the multiclass case the one-vs-rest scheme is used.

Support vector machines (SVM) are also known to perform well on a variety of classification tasks and have been used in the context of hate and offensive speech and abusive language detection in recent years (Malmasi and Zampieri, 2017; Wiegand et al., 2018). We use the rbf kernel. In the multiclass case the one-vs-one scheme is used.

To overcome class imbalances, class weights are adjusted inversely proportional to the train data class frequencies for both classifiers.

As seen in recent challenges focused on offensive language detection in social media like TRAC (Kumar et al., 2018), long-short term memories (LSTM) play an important role and are often used among the best classifiers. Their ability of sequential data iteration and of memorizing recent content across time provides them with a good opportunity not only to analyze individual words, but also to find problem-specific dependencies between them.

3 Models

This chapter describes the development of the models for the submissions and how they perform on the training data.

3.1 Sub-task A

The training data was split into a training set containing 3400 OFF and 7840 NOT labels and a validation set, which consists of the remaining 1000 OFF and 1000 NOT labels. To establish a baseline for this sub-task, we used the features *CHARS*, *WORDS* and *STEMS*, which were the features that were mainly used by the most powerful systems in the TRAC challenge that did not use deep learning. We opted for logistic regression, because it performed better than the SVM. C was set to the default value 1. Table 1 gives an overview of all model results for sub-task A. The baseline approach, denoted as *Baseline A*, reaches a macro F_1 of 0.712 on the validation set.

Our first deep learning model architecture *LSTM A1* uses ELMo context embeddings as inputs into a bidirectional LSTM layer with a size of 64 cells. For classification, we used the hidden states of the LSTM cells generated when passing the last token from each tweet to the LSTM. A fully connected layer was used to compute the two class scores that were normalized using softmax. Its highest macro F_1 value of 0.742 on the validation set was reached after only two epochs, which means that the network tends to overfit very quickly. To overcome this problem, we placed an additional fully connected layer with a dimensionality of 64 and L2 loss between the high-dimensional ELMo output (1000 dimensions) and the LSTM layer to insert a relatively strong regularization loss to the system. We expected this to force the system to look for more universal patterns in the embeddings before passing them to the LSTM layer. While improving the F_1 score by 1%, this architecture, named *LSTM A2*, still started overfitting after two epochs. As a further adjustment against overfitting we used an extra heavy dropout layer with a 70% dropout between the ELMo embeddings and the fully connected layer. We also tried other dropout rates (30%, 40%, 50%, 60%, 80%, 90%), but 70% worked best. A higher percentage did not yield meaningful results. Through this modification of the architecture, *LSTM A3*, the overfitting was delayed until the 16th epoch. The F_1 score also showed a slight improvement.

System	F_1 (macro)	Accuracy
Baseline A	0.712	0.718
LSTM A1	0.742	0.745
LSTM A2	0.753	0.753
LSTM A3	0.759	0.759
LSTM A4	0.729	0.731
LSTM A5	0.764	0.764
LSTM A6	**0.767**	**0.768**
Feature Union A	0.760	0.760

Table 1: Overview of the performance by different models for sub-task A

Comparing non-contextualized to contextualized word embeddings, the developed architecture was also evaluated with GloVe (*LSTM A4*) as well as with a vector concatenation of GloVe and ELMo Embeddings *LSTM A5*. It turned out that ELMo embedding inputs alone exceed the GloVe embedding inputs in this model and that they work together even better than each one on its own.

Due to the remaining problem of overfitting, we used the additional data described in Section 3.1. Pre-training the network with this additional data and then training it with our training set, a small but no significant improvement was achieved. With a macro F_1 of 0.767, this model (*LSTM A6*) shows the best performance on the validation set and was therefore used for our first submission. We used a learning rate of 0.001 and a batch size of 64 for all models. The number of epochs was chosen dependent on the stagnating progress on the validation set.

We used logistic regression with the default $C = 1$ for the second submission system. As feature representation for each tweet, we combined *CHARS*, *WORDS* and *STEMS* with the output of the already trained LSTM of *LSTM A6* as 128-dimensional fixed features. This system *Feature Union A* could not surpass *LSTM A6*, but achieves nevertheless a F_1 of 0.760. It should also be noted, that the χ^2 feature selection could reduce the features used in this feature union to only four of them without affecting the accuracy of the classification. All of them derived from the LSTM output features. The full architecture of this model, including all the models described earlier in this section, is illustrated in Figure 1.

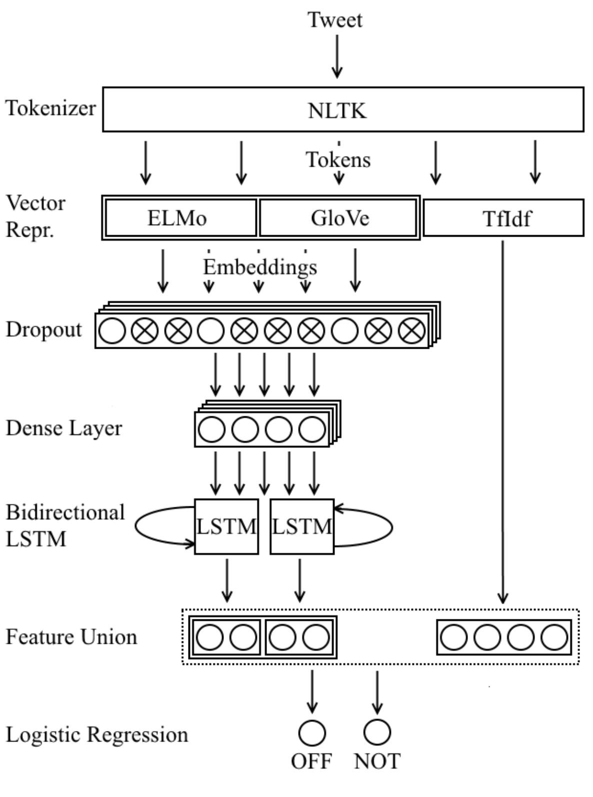


Figure 1: Model architecture for the *Feature Union A* system

3.2 Sub-task B

System	F_1 (macro)	Accuracy
Baseline B	0.615	**0.869**
LSTM B1	0.602	0.745
LSTM B2	0.628	0.809
Feature Union B	**0.653**	0.825

Table 2: Overview of the performance by different models for sub-task B

In sub-task B we split the data into 80% training and 20% validation set. We trained the same baseline model as in sub-task A, which reached a macro F_1 of 0.615 on the validation set and forms the baseline for the list of all models build for this sub-task shown in Table 2.

For our first deep learning approach on this sub-task, we used the same structure as in *LSTM A6*, but with smaller layer sizes as another way to reduce overfitting, because the training data given for this sub-task was even smaller than for sub-task A. The dense layer between the dropout layer and the bidirectional LSTM consists of 8 and the LSTM of 32 cells. This system named *LSTM B1* achieves 0.623 on the validation set.

Pseudo Labeling was used on the additional data described in Section 3.1 to generate the missing labels for this task. For this, we first labeled the additional data using *LSTM B1*, which had already been trained on the training data. Then the resulting labels are used to extend the training data. In addition, we have focused only on the labels which softmax class score was higher than 0.7 for the predicted label to reduce noisy labels. This results in about 2000 UNT and 17500 TIN labeled tweets. The resulting system named *LSTM B2* reaches a macro F_1 of 0.628 on the validation set.

As first classifier *Feature Union B*, we used the same structure as in sub-task A and combined *CHARS*, *WORDS* and *STEMS* and the output of *LSTM B2* as feature input. The value for C in the logistic regression was selected using grid search and was set to 0.51. Together these features improved the macro F_1 even further and reached our best result 0.653 for sub-task B on the validation set.

For the second submission, we trained additional four models of *LSTM B2* and we used the *Feature Union B* from the first submission as a fifth classifier. We then performed a majority vote on all five resulting labels.

All deep learning models in sub-task B are trained with a learning rate of 0.001 and a batch size of 32 for all models. The number of epochs was defined in the same way as in sub-task A.

3.3 Sub-task C

The evaluation for sub-task C is done on a 80%-20% train-validation split making sure that the test data set contains a 20% proportion of each class. Additionally, the train data was enriched with the trial data, where all occurrences of class ORG were replaced by class OTH. Deep learning was not used for the implementation of this sub-task due to the small amount of training data. To select the best feature sets for the used classifiers, logistic regression and SVM all combinations were tested including grid search to find the best parameter values for C and γ, respectively.

Class	Features	Prec.	Recall	F_1
GRP	ELMo	0.566	**0.679**	0.617
	ELMo+NE	0.573	0.674	0.62
	ELMo+ND	0.567	0.61	0.614
	ELMo+NE+ND	0.58	0.674	0.623
	all	**0.582**	0.674	**0.625**
IND	ELMo	0.820	0.789	0.804
	ELMo+NE	0.821	0.795	0.808
	ELMo+ND	0.567	0.67	0.614
	ELMo+NE+ND	0.823	0.795	0.809
	all	**0.825**	**0.798**	**0.811**
OTH	ELMo	0.360	0.237	0.286
	ELMo+NE	0.365	0.250	0.297
	ELMo+ND	0.370	0.263	0.308
	ELMo+NE+ND	0.375	0.276	0.318
	all	**0.386**	**0.29**	**0.331**
macro	ELMo	0.582	0.568	0.57
	ELMo+NE	0.587	0.573	0.575
	ELMo+ND	0.586	0.574	0.575
	ELMo+NE+ND	0.593	0.582	0.584
	all	**0.598**	**0.587**	**0.589**

Table 3: Overview of feature impact on the logistic regression system with $C = 0.011$

Logistic regression classifies best on the validation set using $C = 0.011$ and the combination of the features *WORDS*, *NE*, *SPC*, *DEP*, *SPNR*, *ND* and the ELMo embeddings trained for sub-task B achieving a macro F_1 of 0.589. Table 3 gives an overview of the model performance for the three classes. The features are broken down by influence, where *all* denotes the entire feature set used. ELMo is the most important feature here and gives a macro F_1 of 0.57. The further two percent are largely due to the use of named entities and the noun distribution. The additional features (*WORDS*, *SPC*, *DEP*, *SPNR*) have only little influence of the classification. A closer look reveals that the class performances differ widely depending on class size. Underrepresented classes benefit the most from adding other features to the embeddings, OTH in particular: Precision and recall increase by two and five percent, which leads to a rise in F_1 of about four percent.

Class	Features	Prec.	Recall	F_1
GRP	ELMo	0.537	**0.729**	0.618
	ELMo + SPNR	0.554	0.715	0.625
	all	**0.554**	0.715	**0.625**
IND	ELMo	0.871	0.706	0.78
	ELMo + SPNR	0.873	0.706	0.781
	all	**0.874**	**0.708**	**0.782**
OTH	ELMo	0.296	0.342	0.317
	ELMo + SPNR	0.317	0.434	0.367
	all	**0.32**	**0.434**	**0.369**
macro	ELMo	0.568	0.592	0.572
	ELMo+SPNR	0.582	0.618	0.591
	all	**0.583**	**0.619**	**0.592**

Table 4: Overview of feature impact on the SVM system with $C = 6$, $\gamma = 0.0001$

Table 4 depicts the results for the best SVM model, using a feature set of *WORDS*, *DEP*, *SPNR*, *ND* and the ELMo embeddings trained for sub-task B. The best parameter values we found are $C = 6$ and $\gamma = 0.0001$ and lead to a macro F_1 of 0.592 on the validation set. As in the previous setting, ELMo is the key feature and achieves 0.572 macro F_1. The union with feature *SPNR* leads to 0.591 macro F_1. As with logistic regression, the addition has a positive effect on the less common classes.

In addition, we use a majority vote as third submission. For this, we build a voting classifier out of the previously developed logistic regression and SVM classifiers and a further logistic regression classifier which uses *CHARS*, *WORDS* and *STEMS* and the output of *LSTM B2* as feature input. It should be noted, that the LSTM has not been re-trained on the sub-task C data. Subsequently, a majority decision is taken and in case of doubt, the label prediction of the latter logistic regression classifier is given preference.

4 Results

After the previous description of the submission models, an overview of the test results follows in this section.

4.1 Sub-task A

In order to prepare the *LSTM A6* system for submission, we re-trained it with the complete available training data. This classifier reached a macro F_1 of 0.768, as shown in Table 5 on the test set, which is comparable to the result reached on the validation set in Table 1.

As a second submission we chose the *Feature Union A* classifier, but took the output of the LSTM from the first submission for the LSTM output features in the feature union instead. With 0.742, it reached a somewhat lower macro F_1 on the test set than on our validation set as shown in comparison of Table 1 and Table 5.

System	F1 (macro)	Accuracy
All NOT baseline	0.419	0.721
All OFF baseline	0.218	0.279
LSTM A6	**0.768**	**0.807**
Feature Union A	0.742	0.788

Table 5: Test results for sub-task A

4.2 Sub-task B

The *Feature Union B* system reached a macro F_1 score of 0.671. Comparing the results from Table 2 and Table 6, the classifier works equally well on the train and on the test data.

The majority vote classifier achieved our best result with a macro F_1 score of 0.719 (Table 6). This result is remarkably good which we think is caused by a high robustness resulting from the different strengths of the incorporated classifiers.

System	F1 (macro)	Accuracy
All TIN baseline	0.470	0.888
All UNT baseline	0.101	0.113
Feature Union B	0.671	**0.871**
Majority Vote B	**0.719**	0.850

Table 6: Test results for sub-task B

4.3 Sub-task C

On the test data, all systems perform inferior than on the validation set. The best macro F_1 is obtained by the SVM having 0.571, followed by the logistic regression with 0.551 and then the voting classifier with 0.539. Overall, the performance is fairly stable compared to the validation set and the fact that no cross validation was used in the model selection process.

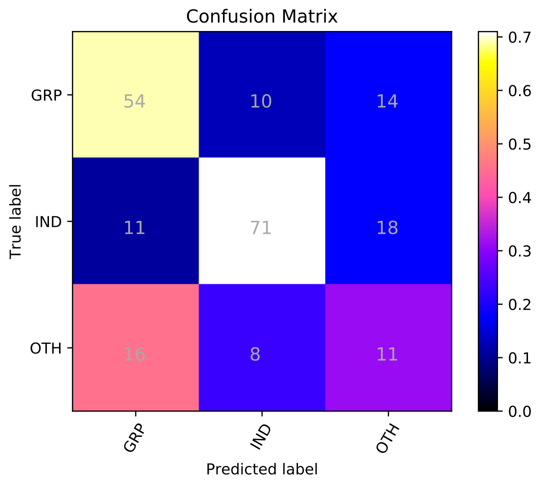


Figure 2: Confusion matrix for sub-task C and the best performing model SVM

Figure 2 shows the confusion matrix for the best performing system. As can be seen, most tweets of GRP and IND are classified correctly, whereas almost half of the OTH tweets are incorrectly recognized as GRP. On the one hand, the rare class occurence in the training seems to obstruct the classifier in learning. On the other hand, it was challenging for us as humans to differentiate between the classes GRP and OTH in the data set: For instance "Liberals are mentally ill!" is labeled as GRP, whereas "@USER republicans/conservatives are the most disgusting people" has label OTH.

5 Conclusion

We showed that contextualized embeddings work well in the context of offensive language identification and the two different categorization tasks. Other language and linguistic features could deliver only small improvements.

The main problems in using deep learning are the model size and the dimensionality and number of parameters resulting in fast overfitting. We plan to address this problem through strategies to reduce model size (primarily in reducing the high-dimensional ELMo output without loosing relevant information before connecting it to the less high-dimensional LSTMs using a fully connected layer), or to include more data when available. Another consideration for sub-task A would be the use of the majority vote model from sub-task B, which showed good results. For sub-task C, the development of deep learning models would be interesting.

References

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2015. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org*, 1(2).

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017a. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017b. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 467–472.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

John Ashworth Nelder and Robert WM Wedderburn. 1972. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3):370–384.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christoper Manning. 2014. Glove: Global vectors for word representation. volume 14, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations.

Martin F Porter, Richard Boulton, and Andrew Macfarlane. 2002. The english (porter2) stemming algorithm. *Retrieved*, 18:2011.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Hope at SemEval-2019 Task 6: Mining social media language to discover offensive language

Gabriel-Florentin Pătraș[1], **Diana-Florina Lungu**[1]
Daniela Gîfu[1,2,3], **Diana Trandabăț**[1]
[1]Alexandru Ioan Cuza University of Iași, Romania
[2]Institute of Computer Science of the Romanian Academy, Iași Branch, Romania
[3]Cognos Business Consulting S.R.L., Romania
{patras.gabriel.florentin, lungu.diana.florina, daniela.gifu, dtrandabat}@info.uaic.ro

Abstract

User's content share through social media has reached huge proportions nowadays. However, along with the free expression of thoughts on social media, people risk getting exposed to various aggressive statements. In this paper, we present a system able to identify and classify offensive user-generated content.

1 Introduction

With the constant spread of social media, users are spending increasing amounts of time on various social networking sites aiming to connect with peers, to share information or common interests. While users benefit from their use of social media by interacting with and learning from others, they are also at the risk of being exposed to large amounts of offensive contents.

Considering that people are negatively affected by harmful contents, detecting online offensive language to protect users online safety becomes an urgent task. To address concerns on people's access to offensive content over the internet, social media administrators often need to manually review online texts to detect and delete offensive materials. However, manually reviewing and identifying offensive messages is a highly human and time consuming task. Some automatic content filtering software packages have been developed to detect and filter offensive WebPages or paragraphs, mostly word-based approaches.

The "OffensEval: Identifying and Categorizing Offensive Language in Social Media" task at the SemEval 2019 competition (Zampieri *et al.*, 2019a) focuses on detecting and classifying offenses, pervasive in social media.

In this paper, we present a system able to identify whether a tweet is abusive language or not, and if abusive, if it is offensive or not. We trained a model to differentiate between these categories and then analyzed the results to better understand how we can improve the system.

The rest of the paper is organized as follows: section 2 presents other projects related to offensive language identification, section 3 presents the project's data set and methods, section 4 presents the results we have obtained and a short analysis, followed by our last point represented by section 5 with the conclusions.

2 Related Work

This topic has attracted significant attention in recent years, evidenced by increasing number of recent publications and a several scientific events such as ALW and TRAC workshops.

Offensive language is often subdivided into various intercalated categories, since different subtasks have been grouped under this label. One of the most analyzed such language is "hate speech", i.e. discriminative remarks, such as the racist or sexist ones (Norbata *et al.*, 2016).

Based on work on hate speech, cyberbullying and online abuse, Waseem *et al.*, 2017 proposses a typology that captures central similarities and differences between subtasks and discuss its implications for data annotation and feature construction. Additionally, Waseem *et al.* (2017) emphasize the practical actions that can be taken by researchers to best approach their abusive language detection subtask of interest.

Lexical detection methods for the offensive language tend to have low precision because they fail to classify messages not containing listed offensive terms. On the other hand, various machine learning methods are used in the literature, from Logistic regression, Naïve Bayes, Decision Trees, Random forests, SMVs to neural networks. Previous analysis of hate speech modeling (Schmidt and Wiegand, 2017) shows that there is a too wide range of features used, and a more advanced feature relevance analysis was needed (Waseem *et al.*, 2017).

A first shared task on aggression identification aiming to classify aggressive speech into overt, covert or no aggression was held at the TRAC Workshop collocated with COLING 2018 (Kumar *et al.*, 2018). 130 teams registered to participate in the task, 30 teams submitted their test runs and 20 teams sent their system description paper, which are included in the TRAC workshop proceedings.

The problem of distinguishing general profanity from hate speech is not a trivial task (Malmasi and Zampieri, 2018) and requires features that capture a deeper understanding of the text not always possible with surface grams.

3 Data set and Methods

The data set for SemEval 2019 task 6 was formed from 14100 tweets, 13240 training instances, retrieved from social media and distributed in tab-separated format and 860 tweets for testing (Zampieri *et al.*, 2019b). Using this data set, we were able to identify offense, aggression and hate speech in user generated content.

This section presents our approach for the different subtask, for each submission we uploaded.

Sub-task A: Offensive language identification

Submission 1. We analyzed the training data to identify specific words or expressions for offensive, respectively non-offensive tweets. Based on these expressions, we crafted a set of rules consisting in exact or partial matches of these expressions in the test corpus. Tweets that have complied with these rules have been annotated as offensive. Tweets containing such expression only in a negated form were annotatd as non-offensive. The rest of the tweets were randomly classified in offensive or non-offensive. The application code was written in the Java programming language and the results are presented in Table 1.1.

Submission 2. We created a lexicon based on two lists of words of offensive lexicons[1], freely available online, along with the list resulted from the analysis of the set of training tweets, as described above. Using these offensive words or expressions, we developed patterns and we classified the tweets in offensive tweets and non-offensive tweets. If the tweet was containing at least one word from the lists, it means that the tweet is offensive, otherwise the tweet would be considered not offensive. The results are presented in Table 1.2.

Submission 3: For this submission, we used the same lists of offensive words obtained from external sources, along with the list of offensive words found in the training data, but we put a restriction on the size of the words (more than 4 letters). This constraint was considered due to the fact that we noticed that they introduced noise in the non-offensive tweets. Additionally, we used WordNet to obtain the synonyms of the words we had in our lists. The results are presented in Table 1.3.

Sub-task B: Automatic categorization of offense types

Submission 1: We tokenized the tweets annotated with targeted offensive words and collected different lists of cue words. Additionally, we noticed that if the tweet contained a proper name towards the middle of the sentence, the tweet was marked as a targeted tweet; otherwise it was marked as an untargeted tweet. We used this restriction and made the first submission, with the results presented in Table 2.1.

Submission 2: For the second submission, we tokenized the test tweets and checked if those words were found in the list of pronouns[2]. If a tweet was containing a pronoun from that list, then that tweet was marked as a targeted offensive one, otherwise it was marked as an untargeted offensive tweet. The results are presented in Table 2.2.

[1] One available at the GitHub repository for the paper (Davidson *et al.*, 2017) and one from Luis von Ahn (2018), consisting on English terms that could be found offensive on websites.

[2] https://www.really-learn-english.com/list-of-pronouns.html

Submission 3: We separated the tweets in words and counted how many words begin with a capital letter. We didn't take into consideration the "#" (hashtags) and @(@USER) because the vast majority were written with a capital letter. If a tweet was containing at least 2 words with capital letter, then the tweet was marked as being a targeted offensive tweet, otherwise was marked as an untargeted offensive tweet. The results are presented in Table 2.3.

Sub-task C: Offense target identification

We created two lists with pronouns. One list was used for the personal pronouns in singular for and the second one for the personal pronouns in plural. Therefore, we obtained 3 scenarios:

- If the tweet contains a personal pronoun from the singular pronoun list, then the tweet is marked IND.
- If the tweet contains a personal pronoun from the plural pronoun list, the tweet is marked GRP.
- If the tweet does not contain any pronouns from the above lists then the tweet is marked as OTH. The results are presented in Table 3.

4 Results

Below are the results for each individual level using the test set. We report Precision (P), Recall (R), and F-measure (F) for each baseline on all classes along with weighted averages and Macro-F1. The result for sub-task A are presented in table 1, the results for sub-task B are presented in table 2 and the results for sub-task C are presented in Table 3.

Sub-Task A: Offensive language identification

	P	R	F	Samples
NOT	0.7398	0.4952	0.5932	620
OFF	0.2966	0.5500	0.3854	240
Avg./ Total	0.6161	0.5105	0.5352	860

Table 1.1: Results Sub-Task A – Submission 1.

	P	R	F	Samples
NOT	0.7876	0.5323	0.6352	620
OFF	0.7324	0.6292	0.4435	240
Avg./ Total	0.6634	0.5593	0.5817	860

Table 1.2: Results Sub-Task A – Submission 2.

	P	R	F	Samples
NOT	0.7718	0.6984	0.7333	620
OFF	0.3746	0.4667	0.4156	240
Avg./ Total	0.6610	0.6337	0.6446	860

Table 1.3: Results Sub-Task A – Submission 3.

Sub-Task B: Automatic categorization of offense types

	P	R	F	Samples
TIN	0.8571	0.4789	0.6145	213
UNIT	0.0826	0.3704	0.1351	27
Avg./ Total	0.7700	0.4667	0.5605	240

Table 2.1: Results Sub-Task B – Submission 1.

	P	R	F	Samples
TIN	0.9091	0.4695	0.6192	213
UNIT	0.1308	0.6296	0.2166	27
Avg./ Total	0.8215	0.4875	0.5739	240

Table 2.2: Results Sub-Task B – Submission 2.

	P	R	F	Samples
TIN	0.9211	0.3286	0.4844	213
UNIT	0.1280	0.7778	0.2199	27
Avg./ Total	0.8318	0.3792	0.4547	240

Table 2.3: Results Sub-Task B – Submission 3.

Sub-Task C: Offense target identification

	P	R	F	Samples
GRP	0.3333	0.0128	0.0247	78
IND	0.4815	0.3900	0.4309	100
OTH	0.1860	0.6857	0.2927	35
Avg./ Total	0.3787	0.3005	0.2595	213

Table 3: Results Sub-Task C.

5 Conclusions

The offensive language in social media commonly comes from an unpleasant condition or something that is disgusting or forbidden. We discussed the challenges in detecting offensive language including the abusive words writing patterns in social media.

This paper presents our system participating at SemEval Task 6. We present simple baseline scores on all classes in all of the three sub-tasks.

In the future, we would like to make a comparison between our system and datasets annotation for similar tasks such as aggression or abusive identification and hate speech detection.

As further work, we have already started to study how to use the datasets for applying deep learning techniques to improve our results, based on word embedding, similar to the work presented in (Badjatiya *et al.*, 2017).

Acknowledgments

This survey was partially supported by a grant of the Romanian Ministry of Research and Innovation, CCCDI – UEFISCDI, project number PN-III-P1-1.2-PCCDI-2017-0818/73PCCDI (ReTeRom), within PNCDI III and by the README project "Interactive and Innovative application for evaluating the readability of texts in Romanian Language and for improving users' writing styles", contract no. 114/15.09.2017, MySMIS 2014 code 119286.

References

Badjatiya, P., Gupta, S., Gupta, M., and Varma, V. 2017. *Deep learning for hate speech detection in tweets*. In Proceedings of the 26th International Conference on World Wide Web Companion, pages 759–760

Davidson, T., Warmsley, D., Macy, M. and Weber, I. 2017. *Automated Hate Speech Detection and the Problem of Offensive Language*. In Proceedings of ICWSM.

Kumar, R., Ojha, A.K., Malmasi, S. and Zampieri, M. 2018. *Benchmarking Aggression Identification in Social Media*. In: Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC), pages 1-11.

Luis von Ahn Research Group (accessed 2018) *Offensive/Profane Word List*, available online at https://www.cs.cmu.edu/~biglou/resources/bad-words.txt?fbclid=IwAR3yLdiB5lsgoQjWIJXgYLPb6Pl4jK-MCT5INw_Lfkfet6A8mvHsB-hyJVY

Malmasi, S., Zampieri, M. 2018. *Challenges in Discriminating Profanity from Hate Speech.* Journal of Experimental & Theoretical Artificial Intelligence, Vol. 30, Issue 2, pages 187-202. Taylor & Francis.

Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y., and Chang, Y. 2016. Abusive language detection in online user content. In Proceedings of the 25th International Conference on World Wide Web. pages 145–153.

Schmidt, A. and Wiegand, M. 2017. A survey on hate speech detection using natural language processing. In Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics, Valencia, Spain, pages 1–10.

Waseem, Z., Davidson, T., Warmsley, D. and Weber, I. 2017. *Understanding Abuse: A Typology of Abusive Language Detection Subtasks*. In: Proceedings of the Abusive Language Online Workshop.

Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., Kumar, R. (2019a) *SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval)* in Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval) 2019.

Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., Kumar, R. (2019b) *Predicting the Type and Target of Offensive Posts in Social Media*, in Proceedings of N AACL 2019.

INGEOTEC at SemEval-2019 Task 5 and Task 6: A Genetic Programming Approach for Text Classification

Mario Graff and **Sabino Miranda-Jiménez** and **Eric S. Tellez**
CONACyT - INFOTEC, Aguascalientes, México
{mario.graff,sabino.miranda,eric.tellez}@infotec.mx

Daniela Moctezuma
CONACyT - CentroGEO, Aguascalientes, México
dmoctezuma@centrogeo.edu.mx

Abstract

This paper describes our participation in HatEval and OffensEval challenges for English and Spanish languages. We used several approaches, B4MSA, FastText, and EvoMSA. Best results were achieved with EvoMSA, which is a multilingual and domain-independent architecture that combines the prediction from different knowledge sources to solve text classification problems.

1 Introduction

Social media platforms, like Twitter and Facebook, are spaces where people interact with others and express themselves; while these platforms encourage free speech, other issues could emerge such as the usage of offensive language that could mock or insult individuals or groups of people. Thus, detecting offenses and misbehavior expressed in text form is interesting to measure the people's feelings and warn them about possible attacks on others such as abusive language, hate speech, cyberbullying, trolling, among others (Waseem et al., 2017).

In order to tackle these text classifications problems, SemEval-2019 proposed two tasks: multilingual detection of hate speech against immigrants and women in Twitter HatEval, task 5 (Basile et al., 2019), and identification and categorization of offensive language in social media OffensEval, task 6 (Zampieri et al., 2019b). In this paper, we present the results from our participating in these two tasks.

The HatEval challenge consists in detecting hate speech for two targets, immigrants and women, in Twitter for Spanish and English languages. There are two subtasks, subtask A is a binary classification where systems have to predict whether a tweet with a given target (immigrants or women) is hateful or not hateful; subtask B is about aggressive behavior and target classification, systems are asked to classify hateful tweets as aggressive or not aggressive, and identify the target harassed (individual or group).

On the other hand, OffensEval challenge consists in determining if a given message has offensive content. It is divided into three subtasks. Subtask A is dedicated to identifying the offensive language, i.e., determine if a message is offensive or not offensive. Subtask B is about categorizing offense types; that is, a tweet containing an insult or threat to someone, or a tweet containing non-targeted profanity and swearing. Finally, subtask C focus on identifying the target, i.e., whether the offensive post is about an individual, a group, or others.

Both HatEval and OffensEval are related tasks to abusive language, Waseem et al. (Waseem et al., 2017) describe tasks on this theme; authors focus their analysis on two primary factors that could guide the modeling of systems: i) language is directed towards a specific individual, entity, or generalized group; ii) the abusive content may be explicit or implicit.

For instance, Schmidt and Wiegand (Schmidt and Wiegand, 2017) present a collection of works on hate speech detection highlighting the features commonly used such as surface-level features. For instance, authors use bag of words (n-grams) and character-level n-grams to attenuate the spelling variation issue on informal text, frequency of URL mentions, punctuation, token lengths, capitalization, among others; word generalization such as topic identification (LDA) and word embeddings (Mikolov et al., 2013); outcomes from sentiment analysis classifiers (for example, samples predicted as negative polarity) as auxiliary evidence of hate for multi-step approaches; usage of lexical resources containing specific negative words (slurs, insults, etc.); linguistic aspects such as parts of speech and syntactic information; knowledge information such as ontologies and taxonomies (ConcepNet, WordNet, etc.).

For both tasks, we use the same approach for final runs. Our approach takes into account several features mentioned above. For example, the effects of character-level n-grams are broadly studied for related tasks in (Tellez et al., 2017b). In particular, text modeling is a crucial factor in our approach; therefore we used the approach presented in (Tellez et al., 2018) that selects the best configuration on the datasets concerned. We also use external knowledge to the given training set to support the classification task; in this sense, our approach named EvoMSA (§2.1) is a stacking system based on genetic programming, and particularly on the use of semantic genetic operators, that focus on sentiment analysis, and, in general, on text classification.

2 System Description

We used our framework based on genetic programming named EvoMSA to evaluate HatEval and OffensEval tasks. EvoMSA is composed of a stack of B4MSA classifiers to produce predictions, and EvoDAG combines the predictions into the final one.

2.1 EvoMSA

EvoMSA[1] (Graff et al., 2018a,b) is a Generic Sentiment Analysis System based on B4MSA and EvoDAG. It is an architecture of two phases to solve classification tasks, see Figure 1. EvoMSA improves the performance of a global classifier combining the predictions of a set of classifiers with different models on the same text to be classified. Roughly speaking, in the first stage, a set of B4MSA classifiers (see Sec. 2.1.1) are trained from several views of the same datasets; datasets provided by SemEval. It creates a decision functions space with mixtures of values coming from different views of knowledge, one coming from B4MSA trained with the training set of the competition (it is used as generic classifier), a lexicon-based model (it only counts affective words: positive and negative, based on several lexicons (Liu, 2017; Albornoz et al., 2012; Sidorov et al., 2013; Perez-Rosas et al., 2012)), an emoji-based space (the sixty-four most probable emoticons for the message) (Graff et al., 2018b), and the output of FastText (Grave et al., 2018) (word embeddings of dimension of 100) trained with the training set. Finally, EvoDAG's inputs are the concatenation of all the decision functions predicted, and EvoDAG produces a final value or prediction. The following subsections describe the internal parts of EvoMSA. The precise configuration of our benchmarked system is described in Sec. 4.

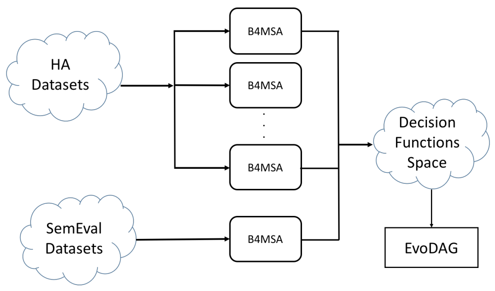


Figure 1: EvoMSA Architecture

2.1.1 B4MSA

B4MSA[2] focus on multilingual sentiment analysis. For complete details of the model see (Tellez et al., 2017a,b). The core idea behind B4MSA is to tackle the sentiment analysis problem as a model selection problem, using a different view of the underlying combinatorial problem, i.e., B4MSA combines a bunch of different text tokenization, text transformations, weighting methods, and internally uses an SVM with a linear kernel to classify. Also, B4MSA takes advantage of several domain-specific particularities like emojis and emoticons and makes explicit handling of negation statements expressed in texts. Nonetheless, EvoMSA avoids the sophisticated use of B4MSA fixing the model for each language in favor of performing an optimization process at the level of the decision functions of several models (Miranda-Jiménez et al., 2017). Table 1 shows text transformation parameters used in our system for English and Spanish languages.

2.1.2 EvoDAG

EvoDAG[3] (Graff et al., 2016, 2017) is a Genetic Programming system specifically tailored to tackle classification and regression problems on very high dimensional vector spaces and large datasets. In particular, EvoDAG uses the principles of Darwinian evolution to create models represented as a directed acyclic graph (DAG). An EvoDAG model

[1]https://github.com/INGEOTEC/EvoMSA

[2]https://github.com/INGEOTEC/b4msa

[3]https://github.com/mgraffg/EvoDAG

has three distinct node's types; the inputs nodes, that as expected received the independent variables, the output node that corresponds to the label, and the inner nodes are the different numerical functions such as sum, product, sin, cos, max, and min, among others. Due to lack of space, we refer the reader to (Graff et al., 2016) where EvoDAG is broadly described.

3 Experimental Settings

As we mentioned, to determine the best configuration of parameters for text modeling, B4MSA integrates a hyper-parameter optimization phase that ensures the performance of the classifier based on the training data. The text modeling parameters for B4MSA were set for all process as we show in Table 1 for English and Spanish languages. A text transformation feature could be binary (yes/no) or ternary (group/delete/none) option. Tokenizers denote how texts must be split after applying the process of each text transformation to texts. Tokenizers generate text chunks in a range of lengths, all tokens generated are part of the text representation. B4MSA allows selecting tokenizers based on n-words, $q-$grams, and skip-grams, in any combination. We call n-words to the popular word n-grams; in particular, we allow to use any combination of unigrams, bigrams, and trigrams. Also, the configuration space allows selecting any combination of character, q-grams, for $q = 1$ to 9. Finally, we allow skip-grams such as $(3, 1)$ and $(2, 2)$, three words separated by one word (gap), and two words separated by two gaps.

We use two baselines B4MSA and the FastText's classifier (Bojanowski et al., 2016) for both contests. FastText represents sentences with a weighted bag of words, and each word is represented as a bag of character n-gram to create text vectors based on word embeddings. Our custom FastText searches automatically the best parameters, e.g., for OffensEval with parameters such as window $size = 9$, learning $rate = 0.01$, $epochs = 10$, size of word $vectors = 10$, minimum and maximum length of character n-grams, 2 and 5, respectively; and some other preprocessing steps such as group numbers and reduce duplicated characters.

3.1 Datasets

SemEval contests provide datasets to train systems for each task. Table 2 presents the data distribution of the HatEval dataset. *Hate* class (HATE) defines tweets that convey hate against immigrants or women; its complement correspond to these messages not having hate content (NO-H), aggressive (AGGR) and no aggressive (NO-A), and target harassed (TARG) as individual and group.

Text transformation	**English (HE)**	**Spanish (HE)**	**English (OE)**
remove diacritics	yes	yes	yes
remove duplicates	yes	yes	yes
remove punctuation	yes	yes	yes
emoticons	group	group	group
lowercase	yes	yes	false
numbers	group	delete	delete
urls	group	none	group
users	group	group	none
hashtags	none	none	none
entities	none	none	none
stemming	yes	yes	yes
Term weighting			
TF-IDF	yes	yes	yes
Entropy	no	no	no
Tokenizers			
n-words	$\{1, 3\}$	$\{1, 2\}$	$\{1, 2, 3\}$
q-grams	$\{3, 5, 9\}$	$\{2, 5, 7, 9\}$	$\{3, 4, 5, 9\}$
skip-grams	$\{(3, 1)\}$	$\{(3, 1)\}$	$\{(3, 1), (2, 2)\}$

Table 1: Example of set of configurations for text modeling, HatEval (HE), and OffensEval (OE)

Table 3 shows the OffensEval data distribution. In Task A, class OFF defines tweets that have offenses or insults; while class NOT describes tweets with no offensive content. Messages with labeled as TIN contain an insult or threat to an entity; UNT defines the opposite. Group (GRP), individual (IND), and others (OTH) classes contain the target of the offensive messages. The OffensEval collection is described in detail in Zampieri et al. (2019a).

DataSet	**NO-H**	**HATE**	**NO-A**	**AGGR**	**NO-T**	**TARG**
training (English)	5,217	3,783	7,441	1,559	7,659	1,341
development (English)	573	427	796	204	781	219
training (Spanish)	2,643	1,857	2,998	1,502	3,371	1,129
development (Spanish)	278	222	324	176	363	137

Table 2: Statistics of HatEval datasets.

DataSet	**Task A**		**Task B**		**Task C**		
	NOT	OFF	TIN	UNT	GRP	IND	OTH
training	8,840	4,400	3,876	524	1,074	2,407	395
development	243	77	34	39	4	30	2

Table 3: Statistics of OffensEval datasets for English language.

4 Results

We present the results of our approaches for HatEval contest in Table 4 and Table 5. We performed our experimentation on the development dataset provided by HateEval. Table 4 shows the results of task A, given a tweet is hateful or not hateful for English and Spanish languages. In the case of task A, the macro-F1 score is used to measure the performance. Table 5 shows the results of task B, classify tweets as aggressive or not aggressive and the target harassed.

In the case of OffenseEval, Table 6 shows the results for the three task proposed offensive language identification (Task A), categorization of offense types (Task B), and offense target identification (Task C).

We present three system configurations for both tasks. B4MSA uses only the training data provided by the contest as the knowledge base to classify texts, i.e., B4MSA is our baseline, but it is also its outcome is an additional input for our more sophisticated classifier (EvoMSA). FastText generates word embeddings from the provided dataset. We do not use pre-training vectors, using pre-trained vectors did not provide any significant improvement in this case, but increased the complexity of the models and the processing pipeline. EvoMSA (Graff et al., 2018a) combines, using EvoDAG, the output of different text models such as B4MSA, a lexicon-based model, an emoji-space model, and FastText.

As we can see the performance in all results Tables, EvoMSA is systematically better than our other systems; under these circumstances, we decided to use EvoMSA firstly in the evaluation phase. Following the rules of HatEval, only the last run would be valid; therefore we used EvoMSA for this chance. In the case of OffensEval, up to three predictions were allowed on the test dataset, but only the best one was compared with other systems. As we can see, Table 6 shows the performance of our three systems on gold standards; EvoMSA stays ahead in all tasks including the baselines from the contest. The table also shows the performance of two baselines, "All NOT" and "ALL OFF", that correspond to labeling all tweets as NOT or OFF, respectively; similarly, the rest of the tasks have baselines for "All TIN", "All UNT", "All GRP", "ALL IND", and "ALL OTH" labeling strategies.

System	F1	Accuracy
English		
B4MSA	0.736	0.752
EvoMSA	0.736	0.733
FastText	0.728	0.756
Performance on gold standard		
EvoMSA	0.350	0.447
Spanish		
B4MSA	0.812	0.838
EvoMSA	0.821	0.834
FastText	0.822	0.801
Performance on gold standard		
EvoMSA	0.710	0.710

Table 4: Results of HateEval: Task A

5 Conclusions

In this paper was presented our solution for HatEval and OffensEval, two campaigns of SemEval 2019. We show the competitiveness of our approach in both training and test phases. EvoMSA and B4MSA are designed to be multilingual and language and domain independent as much as possible. For the training step, we used extra knowledge from datasets out of any specific emotion of the contests, but categories or emotions related to sentiment-analysis information. Our solution performs well in Spanish and some task for English languages; however, there is room for further improvements in performance for tasks in English language using another sort of knowledge for specific domains.

References

Jorge Carrillo De Albornoz, Laura Plaza, and Pablo Gerv. 2012. Language Resources and Evaluation SentiSense : an affective lexicon for sentiment analysis SentiSense : An easily scalable concept-based affective lexicon for sentiment analysis. In *International Conference on Language Resources and Evaluation*, pages 3562–3567, Istanbul, Turkey. European Language Resources Association (ELRA).

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

M. Graff, E. S. Tellez, S. Miranda-Jiménez, and H. J. Escalante. 2016. Evodag: A semantic genetic programming python library. In *2016 IEEE Interna-*

System	Aggressiveness		Hate		Target		Avg-F1
	F1	Accuracy	F1	Accuracy	F1	Accuracy	
English							
B4MSA	0.495	0.814	0.736	0.752	0.659	0.858	0.630
EvoMSA	0.556	0.746	0.736	0.733	0.711	0.851	0.668
FastText	0.536	0.806	0.729	0.752	0.710	0.866	0.658
Performance on gold standard							
EvoMSA	0.515	0.542	0.348	0.445	0.653	0.699	0.506
Spanish							
B4MSA	0.820	0.726	0.785	0.810	0.767	0.880	0.791
EvoMSA	0.755	0.818	0.821	0.834	0.810	0.890	0.795
FastText	0.744	0.818	0.796	0.824	0.791	0.888	0.777
Performance on gold standard							
EvoMSA	0.737	0.765	0.71	0.71	0.816	0.862	0.754

Table 5: Results of HateEval: Task B

System	F1	Accuracy
Task A		
B4MSA	0.767	0.831
EvoMSA	0.774	0.828
FastText	0.741	0.803
Performance on gold standard.		
All NOT baseline	0.419	0.721
All OFF baseline	0.218	0.279
B4MSA	0.729	0.801
EvoMSA	0.731	0.791
FastText	0.697	0.797
Task B		
B4MSA	0.398	0.507
EvoMSA	0.694	0.699
FastText	0.618	0.644
Performance on gold standard.		
All TIN baseline	0.470	0.888
All UNT baseline	0.101	0.113
B4MSA	0.507	0.892
EvoMSA	0.671	0.871
FastText	0.634	0.896
Task C		
B4MSA	0.418	0.833
EvoMSA	0.392	0.611
FastText	0.550	0.806
Performance on gold standard.		
All GRP baseline	0.179	0.366
All IND baseline	0.213	0.470
All OTH baseline	0.094	0.164
B4MSA	0.486	0.653
EvoMSA	0.576	0.676
FastText	0.504	0.639

Table 6: Results of OffensEval: Offensive language identification (Task A), categorization of offense types (Task B), and offense target identification (Task C).

tional Autumn Meeting on Power, Electronics and Computing (ROPEC), pages 1–6.

Mario Graff, Sabino Miranda-Jiménez, Eric S Tellez, and Daniela Moctezuma. 2018a. Evomsa: A multilingual evolutionary approach for sentiment analysis. *arXiv preprint arXiv:1812.02307*.

Mario Graff, Sabino Miranda-Jiménez, Eric Sadit Tellez, and Daniela Moctezuma. 2018b. Evomsa: A multilingual evolutionary approach for sentiment analysis. *CoRR*, abs/1812.02307.

Mario Graff, Eric S. Tellez, Hugo Jair Escalante, and Sabino Miranda-Jiménez. 2017. Semantic Genetic Programming for Sentiment Analysis. In Oliver Schtze, Leonardo Trujillo, Pierrick Legrand, and Yazmin Maldonado, editors, *NEO 2015*, number 663 in Studies in Computational Intelligence, pages 43–65. Springer International Publishing. DOI: 10.1007/978-3-319-44003-3_2.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning Word Vectors for 157 Languages. In *Proceedings of the 11th Language Resources and Evaluation Conference*, pages 3483–3487. Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018).

Bing Liu. 2017. English Opinion Lexicon.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Sabino Miranda-Jiménez, Mario Graff, Eric S Tellez, and Daniela Moctezuma. 2017. INGEOTEC at SemEval 2017 Task 4: A B4MSA Ensemble based on Genetic Programming for Twitter Sentiment Analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, pages 771–776. Association for Computational Linguistics.

Veronica Perez-Rosas, Carmen Banea, and Rada Mihalcea. 2012. Learning Sentiment Lexicons in Spanish. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 3077–3081.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Grigori Sidorov, Sabino Miranda-Jiménez, Francisco Viveros-Jiménez, Alexander Gelbukh, No Castro-Sánchez, Francisco Velásquez, Ismael Díaz-Rangel, Sergio Suárez-Guerra, Alejandro Treviño, and Juan Gordon. 2013. Empirical Study of Machine Learning Based Approach for Opinion Mining in Tweets. In *Proceedings of the 11th Mexican International Conference on Advances in Artificial Intelligence - Volume Part I*, MICAI'12, pages 1–14, Berlin, Heidelberg. Springer-Verlag.

Eric S. Tellez, Sabino Miranda-Jiménez, Mario Graff, Daniela Moctezuma, Ranyart R. Suárez, and Oscar S. Siordia. 2017a. A simple approach to multilingual polarity classification in Twitter. *Pattern Recognition Letters*, 94:68–74.

Eric S. Tellez, Sabino Miranda-Jimnez, Mario Graff, Daniela Moctezuma, Oscar S. Siordia, and Elio A. Villaseor. 2017b. A case study of spanish text transformations for twitter sentiment analysis. *Expert Systems with Applications*, 81:457 – 471.

Eric S. Tellez, Daniela Moctezuma, Sabino Miranda-Jiménez, and Mario Graff. 2018. An automated text categorization framework based on hyperparameter optimization. *Knowledge-Based Systems*, 149:110–123.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

JCTICOL at SemEval-2019 Task 6: Classifying Offensive Language in Social Media using Deep Learning Methods, Word/Character N-gram Features, and Preprocessing Methods

Yaakov HaCohen-Kerner, Ziv Ben-David, Gal Didi,
Eli Cahn, Shalom Rochman, and Elyashiv Shayovitz
Department of Computer Science, Jerusalem College of Technology, Lev Academic Center
21 Havaad Haleumi St., P.O.B. 16031, 9116001 Jerusalem, Israel
kerner@jct.ac.il, benda1237@gmail.com,
galdd8@gmail.com, eli.cahn@gmail.com,
shal.rochman@gmail.com, elyashiv12@gmail.com

Abstract

In this paper, we describe our submissions to SemEval-2019 task 6 contest. We tackled all three sub-tasks in this task "OffensEval - Identifying and Categorizing Offensive Language in Social Media". In our system called JCTICOL (Jerusalem College of Technology Identifies and Categorizes Offensive Language), we applied various supervised ML methods. We applied various combinations of word/character n-gram features using the TF-IDF scheme. In addition, we applied various combinations of seven basic preprocessing methods. Our best submission, an RNN model was ranked at the 25th position out of 65 submissions for the most complex sub-task (C).

1 Introduction

Offensive language is frequent in social media. For instance, ScanSafe's monthly "Global Threat Report" reported that up to 80% of blogs contained offensive contents and 74% included porn in the format of the image, video, or offensive language (Cheng, 2007). There are people that take advantage of the perceived anonymity of computer-mediated communication, using this to write in behavior that many of them would not consider in real life.

Online news and social networking services, online communities, social media platforms, and various computer companies have been investing a lot of effort, time and money to cope with offensive language in order to prevent abusive behavior.

Computational methods are among the most effective strategies to identify various types of aggression, offense, and hate speech in user-generated content (e.g., comments, microblogs, posts, and tweets). Detection of offensive language has been investigated in recent years in various studies (Waseem et al. 2017; Davidson et al., 2017, Malmasi and Zampieri, 2018, Kumar et al. 2018) and various workshops such as ALW (Abusive Language Online) and TRAC (Trolling, Aggression, and Cyberbullying).

In this paper, we describe our submissions to SemEval-2019 task 6 contest. In Task-6, OffensEval, there are three different sub-tasks. Sub-task A deals with offensive language identification. Sub-task B deals with the automatic categorization of offense types. Sub-task C deals with offense target identification.

The report of the OffensEval task is described in Zampieri et al. (2019A) and the description of the OLID dataset that was used for the competition is in Zampieri et al. (2019B).

The structure of the rest of the paper is as follows. Section 2 discusses work related to offensive language in social media, tweet classification, and data preprocessing. Section 3 presents, in general, the task description. In Section 4, we describe the submitted models and their experimental results. Section 6 summarizes and suggests ideas for future research.

2 Background

2.1 Offensive Language in Social Media

In recent years, there has been an increase in the number of studies dealing with Offensive language in social media. Nobata et al. (2016) developed a machine learning based method to detect hate speech on online user comments from two domains. They also built a corpus of user comments annotated accordingly to three subcategories (hate speech, derogatory, profanity). Waseem and Hovy (2016) introduced a list of criteria founded in critical race theory and used them to label a publicly available corpus of more than 16k tweets with tags about both racial and sexist offenses.

A survey on hate speech detection is presented by Schmidt and Wiegand (2017). The authors introduced various NLP methods that were developed in order to detect hate speech. Davidson et al. (2017) presented a multi-class classifier to distinguish between three categories: hate speech, offensive language, and none of these two. The analysis of the predictions and the errors show when we can reliably separate hate speech from other offensive language and when this differentiation is more difficult. Anzovino et al. (2018) built a labelled corpus containing 2,227 misogynous (hate speech against women) tweets and no-misogynous tweets and explored various NLP features and ML models for detecting and classifying misogynistic language.

2.2 Tweet Classification

Sriram et al. (2010) presented a new classification model that uses a small set of domain-specific features extracted from the author"s profile and text. Experimental results showed that the classification accuracy of their model is better than the classification accuracy of the traditional Bag-Of-Words model. Batool et al. (2013) introduced a system that extracts knowledge from tweets and then classifies the tweets based on the semantics of knowledge contained in them. For avoiding information loss, knowledge enhancer is applied that enhances the knowledge extraction process from the collected tweets.

Stance classification of tweets was investigated by HaCohen-Kerner et al. (2017). Given test datasets of tweets from five various topics, they classified the stance of the tweet authors as either in FAVOR of the target, AGAINST it, or NONE. Their algorithm used a few tens of features mainly character-based features where most of them are skip char ngram features. The experimental results showed that this algorithm significantly outperforms the traditional 'bag-of-words' model.

2.3 Data preprocessing

Data preprocessing is an important step in data mining (DM) and ML processes. In tweets, it is common to find typos, emojis, slang, HTML tags, spelling mistakes, irrelevant and redundant information. Analyzing data that has not been carefully cleaned or pre-processed might lead to misleading results.

Not all of the preprocessing types are considered effective in the text classification community. For instance, Forman (2003), in his study on feature selection metrics for text classification, claimed that stop words occurring frequently and are ambiguous and therefore should be removed, However, HaCohen-Kerner et al. (2008) demonstrated that the use of word unigrams including stop words lead to improved text classification results compared to the results obtained using word unigrams excluding stop words in the domain of Hebrew-Aramaic Jewish law documents.

In our system, we applied various combinations of seven basic preprocessing types: C - spelling Correction[1] using a dictionary of containing 479k English words[2], L – converting to Lowercase letters, P – Punctuation removal, S – Stopwords Removal, R – Repeated characters removal, T – sTemming, and M - leMmatizion) in order to employ the best combination.

3 The Competition of Task 6

SemEval-2019 Task 6 consists of three subtasks:

1. Subtask A: Given a tweet, predict whether it contains offensive language or a targeted (veiled or direct) offense or it does not contain offense or profanity.
2. Subtask B: Given a tweet containing offensive language, predict whether it contains an insult or threat to an individual, a group, or others, or contains non targeted profanity and swearing.
3. Subtask C: Given a tweet containing an insult or threat, predict whether the target is an individual or a group of people considered as a unity due to the same ethnicity, gender or sexual orientation, political affiliation, religious belief, or something else, or does not belong to any of the previous two categories (e.g., an organization).

The dataset of Task 6 contains tweets that were annotated using crowdsourcing. The dataset of sub-task A contains 13,240 tweets: 4,404 OFF (Offensive language) tweets (about 33%) and 8,836 NOT (Not Offensive) tweets (about 67%).

[1] https://github.com/phatpiglet/autocorrec.

[2] https://github.com/dwyl/english-words.

The dataset of sub-task B contains 4,400 offensive tweets: 3,876 TIN (Targeted Insult) tweets (about 88%) and 524 UNT (Untargeted) tweets. The dataset of sub-task C contains 3,876 tweets: 2,407 IND (Individual) tweets (about 62%), 1,074 GRP (Group) tweets (about 28%), and 395 OTH (Other) (about 10%). The test data of sub-tasks A, B, and C contain 860, 240, and 213 unlabeled tweets, respectively.

4 The Submitted Models and Experimental Results

We have submitted 17 models: 6 models to task 6-A, 6 models to task 6-B, and 5 models to task 6-C. We applied the Python module called Scikit-learn (Pedregosa et al., 2011) using the TF-IDF scheme called TfidfTransformer[3] and we applied various supervised ML methods with various numbers of n-gram features, skip word/char n-grams (HaCohen-Kerner et al., 2017) and combinations of pre-processing types.

While all teams' submissions in all three sub-tasks of task 6 were ranked according to their F-Measure scores, we were wrong in all these sub-tasks in the sense that we submitted models according to their accuracy scores.

Most of our submitted models were RNN models. Each RNN model was a bidirectional RNN with 4 hidden layers, with different numbers of LSTMs, values of Dropout, and number of vectors of GloVe (Pennington et al., 2014). Additional explanations to our RNN models, which are given in the next paragraphs are mainly based on the explanations given by Nikolai Janakiev in "Practical Text Classification with Python and Keras"[4].

We used the Tokenizer utility class, which converts a text corpus into a list of integers. Each integer maps to a value in a dictionary that encodes the entire corpus, with the dictionary's keys being the vocabulary terms themselves.

We chose to use the Twitter-aware tokenizer, designed to be flexible and easy to adapt to new domains and tasks (e.g., for tweet processing).

We used the word embeddings method. This method represents words as dense word vectors, which are trained, unlike the one-hot encoding which is hardcoded. The word embeddings map the statistical structure of the language used in the corpus. Their aim is to map semantic meaning into a geometric space. This geometric space is then called the embedding space. This method would map semantically similar words close on the embedding space.

There are two options to get such a word embedding. One way is to train the word embeddings during the training of our neural network. The other way is to use a precomputed embedding space that utilizes a larger corpus. Among the most popular methods are GloVe (Global Vectors for Word Representation) developed by the Stanford NLP Group (Pennington et al., 2014) and Word2Vec developed by Mikolov et al. (2013).

GloVe applies a co-occurrence matrix and by using matrix factorization while Word2Vec applies neural networks. Word2Vec is more accurate and GloVe is faster to compute. We used the GloVe method for our model.

4.1 Results of Task 6-A

Table 1 presents the main characteristics and results of our six submitted models to task 6-A. The models are presented in descending order according to their F-measure score on the test set.

[3] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html#sklearn.feature_extraction.text.TfidfTransformer

[4] https://realpython.com/python-keras-text-classification/#author

The first name of the model authors	Pre-proc-essing	Model				Score			
		ML Meth od	N-Gram Features	Additional Features (for RNN only)	FC Layer (for RNN only)	CV	Test Scores		
						Acc.	F-M	Acc.	Rank
JCTICOL-Ziv Ben-David	-	RNN	5000 word unigrams, 100 word bigrams	512 LSTMs, 0.2 Dropout. GloVe: 100d.	Logistic Regression Random Forest SVM-linear SVC (kernel=linear) SVC (kernel=rbf) SVC (DFS[5]=ovo) KNeighbors	0.85	0.74	0.81	49/ 103
JCTICOL-Eli Cahn	CLS	RNN	None	512 LSTMs, 0.2 Dropout. GloVe: 100d.	-	0.77	0.73	0.8	50/ 103
JCTICOL-Gal Didi	-	RNN	5000 word unigrams, 100 word bigrams	512 LSTMs, 0.3 Dropout. GloVe: 100d.	Logistic Regression Random Forest SVM-linear	0.85	0.73	0.79	52/ 103
JCTICOL-Shalom Rochman	L	RNN	None	512 LSTMs, 0.2 Dropout. GloVe: 100d.	-	0.75	0.73	0.81	59/ 103
JCTICOL-Elyashiv Shayovitz	-	RNN	5000 word unigrams, 100 word bigrams	512 LSTMs, 0.2 Dropout. GloVe: 100d.	SVC (kernel=rbf)	0.86	0.72	0.81	62/ 103
JCTICOL-Yaakov HaCohen-Kerner	-	SVM-linear	5000 word unigrams, 200 word bigrams, 100 words trigrams	-	-	0.72	0.72	0.78	67/ 103

Table 1: Results of our 6 models in task-A.

The main results and conclusions that can be derived from Table 1 are as follows:

- The best submitted model is an RNN model with F-measure of 0.74 and accuracy of 0.81 obtaining the 43rd position out of 103 submissions.
- The best combination of N-gram features for this model contains 5000 word unigrams and 100 word bigrams (without any word trigrams).
- In addition, this model used 512 LSTMs and in its FC layer, it used seven different ML methods.
- This model did not use any combination of pre-processing types.
- Simpler RNN models and non RNN models e.g. the SVM-linear model (last row in Table 1) as well as other models that were tested but not submitted, were less successful.

4.2 Results of Task 6-B

Table 2 presents the main characteristics and results of our six submitted models to task 6-B. The models are presented in descending order according to their F-measure score on the test set. It should be noted that the train set for sub-task b contains imbalanced sets of tweets. The number of tweets classified as UNT is 524 (about 12%) while the number of tweets classified as TIN is 3,876 (about 88%). The main results and conclusions that can be derived from Table 2 are as follows:

- Our best submitted model is a SVC - support vector classifier with F-measure of 0.49 and accuracy of 0.85 obtaining the 62nd position

[5] DFS – Decision Function Shape.

out of 75 submissions. This model used a combination of the MPR pre-processing types.

- This model used 10,000 char trigrams where for each character trigram we allow up to a maximum of 7 skipped characters in-between the chosen ones.
- As mentioned before, while the submitted models were ranked according to their F-Measure results, we were wrong and submit models according to their accuracy results.

User	**Pre-processing**	**Model**		**Score**			
		ML Method	**N-Gram Features**	**CV**	**Test Score**		
				Acc.	**Macro-F1**	**Acc.**	**Rank**
JCTICOL-Eli Cahn	MPR	SVC - Support vector classifier	10000 char trigrams with 7 skips	0.87	0.49	0.85	62 / 75
JCTICOL-Ziv Ben- David	L	MLP - Multilayer perceptron	10000 char unigrams with 4 skips	0.87	0.48	0.85	63 / 75
JCTICOL-Gal Didi	MPRS	SVC - Support vector classifier	7000 word bigrams with 0 skips	0.87	0.47	0.82	65 / 75
JCTICOL-Elyashiv Shayovitz	CMPR	LR - Logistic regression	10000 char bigrams with 7 skips	0.87	0.47	0.89	66 / 75
JCTICOL-Yaakov HaCohen-Kerner	CLS	SVC - Support vector classifier	1000 char trigrams with 9 skips	0.87	0.47	0.89	67 / 75
JCTICOL-Shalom Rochman	CMP	RF - Random forest	7000 word unigrams with 0 skips	0.87	0.47	0.81	69 / 75

Table 2: Results of our 6 models in task-B.

4.3 Results of Task 6-C

Table 3 presents the main characteristics and results of our six submitted models to task 6-C. The models are presented in descending order according to their F-measure score on the test set.

The main results and conclusions that can be derived from Table 3 are as follows:

- Our best submitted model is an RNN model with F-measure of 0.53 and accuracy of 0.64 obtaining the 25th position out of 65 submissions.
- The best combination of N-gram features for this model contains 5000 word unigrams and 200 word bigrams (without any word trigrams).
- In addition, this model used 512 LSTMs and in its FC layer it used only the SVC ML method.
- This model did not use any combination of pre-processing types.
- Simpler RNN models and non RNN models such as the SVC-linear model (last row in Table 3) as well as other models that were tested but not submitted to the competition, were less successful.

User	**Pre proc essi ng**	**Model**				**Score**			
		ML Method	**N-Gram Features**	**Additional Features (for RNN only)**	**FC Layer (for RNN only)**	**CV**	**Test Scores**		
						Acc.	**F-M**	**Acc.**	**Rank**
JCTICOL-Gal Didi	-	RNN	5000 word unigrams, 200 word bigrams	512 LSTMs, 0.3 Dropout. GloVe: 200d special for Tweeter	SVC (kernel=linear)	0.88	0.53	0.64	25 / 65
JCTICOL-Ziv Ben-David	-	RNN	8000 word unigrams, 200 word bigrams	512 LSTMs, 0.3 Dropout. GloVe: 200d special for Tweeter	Logistic Regression	0.89	0.51	0.64	33 / 65
JCTICOL-Elyashiv Shayovitz	-	RNN	5000 word unigrams, 200 word bigrams	512 LSTMs, 0.3 Dropout. GloVe: 200d special for Tweeter	-	0.68	0.50	0.67	40 / 65
JCTICOL-Yaakov HaCohen-Kerner	-	RNN	5000 word unigrams, 200 word bigrams	512 LSTMs, 0.3 Dropout. GloVe: 200d special for Tweeter	SVM-linear	0.88	0.49	0.62	42 / 65
JCTICOL-Shalom Rochman	-	SVC_ (kernel= linear)	5000 word unigrams, 200 word bigrams	-	-	0.64	0.42	0.54	58 / 65

Table 3: Results of our 6 models in task-C.

5 Conclusions and Future Research

In this paper, we describe our submissions to three sub-tasks of Task 6 of SemEval-2019 contest. Our system JCTICOL (Jerusalem College of Technology Identifies and Categorizes Offensive Language) includes 17 formal submissions: 6 for sub-task A, 6 for sub-task B, and 5 for sub-task C. We used the TF-IDF scheme and we applied various supervised ML methods with various numbers of n-gram features and combinations of pre-processing types. Our best submission was ranked at the 25th position out of 65 submissions for the most complex sub-task (C).

Future research proposals that may contribute to better classification are as follows. (1) Using additional feature sets such as stylistic feature sets (HaCohen-Kerner et al., 2010B) and keyphrases that can be extracted from the text corpora (HaCohen-Kerner et al., 2007); (2) Using acronym disambiguation (e.g., HaCohen-Kerner et al., 2010A), i.e., selecting the correct long form of the acronym depending on its context will enrich the tweet's text; and (3) Using other deep learning models.

Acknowledgments

This research was partially funded by the Jerusalem College of Technology, Lev Academic Center.

References

Rabia Batool, Asad Masood Khattak, Jahanzeb Maqbool, and Sungyoung Lee 2013. Precise tweet classification and sentiment analysis. In Computer and Information Science (ICIS), 2013 IEEE/ACIS 12th International Conference on (pp. 461-466). IEEE.

Jacqui Cheng. 2007. Report: 80 percent of blogs contain "offensive" content, in ars technica. vol. 2011.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In Eleventh International AAAI Conference on Web and Social Media, pages 512-515.

George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. Journal of machine learning research, 3(Mar), 1289-1305.

Yaakov HaCohen-Kerner, Ittay Stern, David Korkus, and Erick Fredj. 2007. Automatic machine learning of keyphrase extraction from short html documents written in Hebrew. *Cybernetics and Systems: An International Journal*, *38*(1), 1-21.

Yaakov HaCohen-Kerner, Dror Mughaz, Hananya Beck, and Elchai Yehudai 2008. Words as classifiers of documents according to their historical period and the ethnic origin of their authors. Cybernetics and Systems: An International Journal, 39(3), 213-228.

Yaakov HaCohen-Kerner, Ariel Kass, and Ariel Peretz. 2010A. HAADS: A Hebrew Aramaic abbreviation disambiguation system. Journal of the American Society for Information Science and Technology, 61(9), 1923-1932.

Yaakov HaCohen-Kerner, Hananya Beck, Elchai Yehudai, and Dror Mughaz. 2010B. Stylistic feature sets as classifiers of documents according to their historical period and ethnic origin. Applied Artificial Intelligence, 24(9), 847-862.

Yaakov HaCohen-Kerner, Ziv Ido, and Ronen Ya'akobov. 2017. Stance classification of tweets using skip char Ngrams. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 266-278). Springer, Cham.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In Proceedings of the First Workshop on Trolling, Aggression, and Cyberbullying (TRAC-2018) (pp. 1-11).

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in discriminating profanity from hate speech. Journal of Experimental & Theoretical Artificial Intelligence, 30(2), 187-202.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In: Proceedings of the 25th International Conference on World Wide Web, pp. 145–153. International World Wide Web Conferences Steering Committee.

Fabian Pedregosa, Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R. and Dubourg, V. and Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), 2825-2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics, Valencia, Spain, pages 1–10.

Bharath Sriram, David Fuhry, Engin Demir, Hakan Ferhatosmanoglu. 2010. Short text classification in twitter to improve information filtering. In Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval (pp. 841-842). ACM.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding abuse: A typology of abusive language detection subtasks. arXiv preprint arXiv:1705.09899.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019A. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval).

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019B. Predicting the Type and Target of Offensive Posts in Social Media. In Proceedings of NAACL.

jhan014 at SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media

Jiahui Han
University of Ottawa
jhan014@uottawa.ca

Shengtan Wu
Jackson State University
shengtan.wu
@students.jsums.edu

Xinyu Liu
Purdue University
liu1957@purdue.edu

Abstract

In this paper, the team jhan014 presents two methods to identify and categorize the offensive language in Twitter. In the first method, we develop a deep neural network consisting of bidirectional recurrent layers with Gated Recurrent Unit (GRU) cells and fully connected layers. In the second method, we establish a probabilistic model, modified sentence offensiveness calculation (MSOC) to evaluate the sentence offensiveness level and target level according to different sub-tasks. Based on task results, We evaluate the performance of each method based on F1 score and analyze the advantages and disadvantages of these two methods with the type I error and type II error. In conclusion, deep neural network behaves well in all subtasks but has more type I error and fails to categorize subclasses with minor data or less character, while MSOC model does better in target categorizing but has more type II error in offensive identifying.

1 Introduction

With the popularity of social media like Twitter, offensive language has become a serious problem(Zampieri et al., 2019b) on these media platforms. People have to face with abusive behavior from others in social media from time to time. To solve this problem, finding a method to identify and categorize offensive languages is an urgent need.

In this paper, two different methods, deep learning method and modified sentence offensiveness calculation method, are used to categorize the type and target of offensive language and the difference of results are revealed and analyzed.

2 Related Work

Deep learning method: Deep learning methods are widely used in natural language processing (Liu et al., 2016). Models like Recursive neural network are commonly used to identify if a sentence contain certain emotion. In our work, a deep neural network with GRU layers and all connection layers is built.

Offensiveness Content Filtering: Offensive language targets can be understood through the sentence structure (Silva et al., 2016) or lexical analysis (ElSherief et al., 2018). We take both sentence structure and the offensiveness level of words into consideration. Furthermore, we also concentrate on the special punctuation (like @ and #) in online social media.

3 Methodology and Data

3.1 Deep Neural Network

In the offensive language detection task, we developed a deep neural network based system with binary cross-entropy output.

System Design The system consists of bidirectional recurrent layers with Gated Recurrent Unit(GRU) cells and fully connected layers (Chung et al., 2014). Because the output of the last time-step is used as the embedding of a sentence, we conduct zero padding in the beginning of each sequence to construct the feature matrix. The system architecture is shown in Table 1.

Optimization Steps Parameters in both RNN layers and Dense layers are initialized by Xiaver initialization method (Glorot and Bengio, 2010). The model is optimized by Adam optimization method with 0.01 learning rate. While training the

Layer Name	Output Dimension	Parameter #
Embedding	100	1000000
GRU	128	63360
GRU	128	74112
GRU	128	74112
GRU	128	74112
Dense	256	33024
Dense	128	32896
Dense	64	8256
Dense	32	2080
Dense	2	66

Table 1: System Architecture

neural network, an early stopping method with 2-iteration tolerance is applied to monitor the process. Once the early stopping method is triggered, we manually lower the learning rate by $1/10$ to overcome the vibration and search for a smaller minimum loss.

3.2 Modified Sentence Offensiveness Calculation

Based on the sentence offensiveness calculation method in this paper(Chen et al., 2012), we develop a model to evaluate the sentence offensiveness.

Offensiveness Dictionary Construction We can always find pejoratives, profanities, or obscenities in offensive twitters. Strongly profanities are always undoubtedly offensive when at users or related to some topics (like #) directly; but there are many other weakly pejoratives and obscenities that may also be offensive.
Word offensiveness is defined(Chen et al., 2012) as: for each offensive word, w, its offensiveness

$$O_w = \begin{cases} a_1 & \text{if w is a strongly offensive word} \\ a_2 & \text{if w is a weakly offensive word} \\ 0 & \text{otherwise} \end{cases}$$

where $0 < a_1 < a_2 < 1$, for the offensiveness of strongly offensive words is higher than weakly offensive words.

Syntactic Intensifier Detection We also built the syntactic features by an intensifier(Zhang et al., 2009). In a sentence, words syntactically related to offensive word, w, are categorized in an intensifier set, $i_w = \{c_1, \ldots, c_k\}$, for each word c_j, its intensify value, d_j, is defined as

$$d_j = \begin{cases} b_1 & \text{if } \mathrm{c_j} \text{ is @ or \#} \\ b_2 & \text{if } \mathrm{c_j} \text{ is an offensive word} \\ 1 & \text{otherwise} \end{cases}$$

where $0 < b_1 < b_2 < 1$, for offensive words used to describe users are more offensive than the words used to describe other offensive words. Thus, the value of intensifier, I_w, for offensive word, w, can be calculated as $\sum_{j=1}^{k} d_j$.

Sentence Level Offensiveness Value Consequently, the offensiveness value of sentence, s, becomes a determined linear combination of words' offensiveness

$$O_s = \sum O_w I_w$$

From the training data, we learn two thresholds θ_1, θ_2. For each sentence, s, we apply these two values

$$P(s = OFF) = \begin{cases} 1 & \text{if } \mathrm{O_s} > \theta_2 \\ \frac{O_s - \theta_1}{\theta_2 - \theta_1} & \text{if } \theta_1 \le \mathrm{O_s} \le \theta_2 \\ 0 & \text{if } \mathrm{O_s} < \theta_1 \end{cases}$$

If the offensiveness value is greater than θ_1, the language will be seen as offensive, while if it is smaller than θ_2 then the language will be not offensive. Otherwise, the result will follow a probabilistic distribution.

When solving other sub-tasks, this method can also be used with changing the dictionary and redefine the target words list.

3.3 Data

We use the datasets in Zampieri et al. (2019a) and apply following methods to preprocess or transform the data.

3.3.1 Preprocessing

The raw twitter data is preprocessed by a data pipeline. All the information which has nothing to do with word vectors such as stop words and emojis are stripped and the output of the pipeline are lower-case stemmed word sequencies.

3.3.2 Word Embedding

A word embedding step is applied to transform the text into numerics for deep neural networks. 100-dimensional Global Vectors(GloVe) word embeddings trained with twitter data are applied in this study considering the trade-off between performance and efficiency of the training process (Pennington et al., 2014). We also explore embedding layers in this study and the pretrained embedding out-performs the embedding layer because of the immense amount of information brought by GloVe's training set with 27-billion tweets.

4 Results

4.1 Sub-task A - Offensive language identification

When identifying whether a sentence is offensive or not, two methods show great difference while the accuracy and F1-score are close (see Table 2). In RNN method, there is more type I error (see Figure 1) which means the model classifies some non-offensive sentences as offensive ones. Since origin dataset is unbalanced, the neural network may not have enough non-offensive training examples to learn. Consequently, it cannot catch the feature and structure of the non-offensive sentences.
In MSOC method, this problem is improved. Due to fixed human defined offensiveness dictionary, the non-offensive sentence is not easily misclassified as offensive one. However, since there are still some offensive words appeared in dataset that are not defined in the dictionary, there is still much type II error (see Figure 2).

System	F1 (macro)	Accuracy
All NOT	0.4189	0.7209
All OFF	0.2182	0.2790
RNN	0.6899	0.7395
MSOC	0.6761	0.7895

Table 2: Results for Sub-task A.

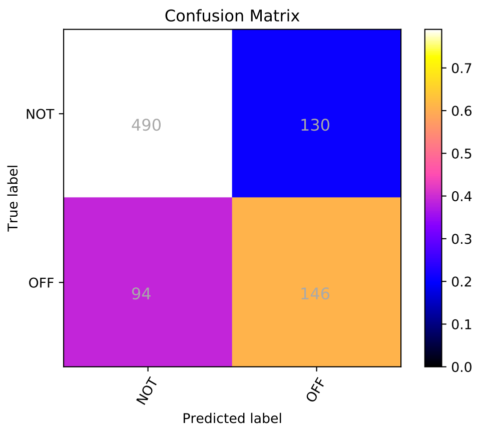


Figure 1: Sub-task A,RNN method

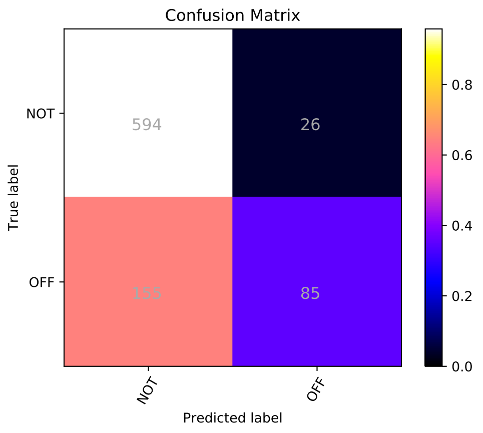


Figure 2: Sub-task A,MSOC method

4.2 Sub-task B - Automatic categorization of offense types

The behavior of MSOC method defeats RNN method from all aspects (see Table 3 and Figure 3, 4) when categorizing the types of offense. This is because usually targeted offensive language have different sentence structure with untargetted ones, this make it a really high accuracy approach to categorize offensive type. In details, a target sentence always contains third-person pronouns like him her it them. And in most target tweets, the sentence has some special punctuation like @ and also related to some hot topics #.

System	F1 (macro)	Accuracy
All TIN	0.4702	0.8875
All UNT	0.1011	0.1125
RNN	0.6153	0.8667
MSOC	0.7545	0.925

Table 3: Results for Sub-task B.

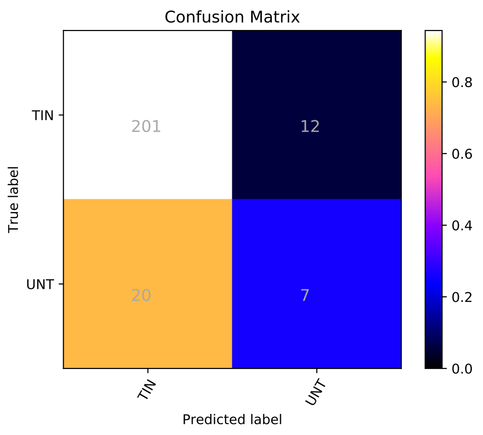


Figure 3: Sub-task B,RNN method

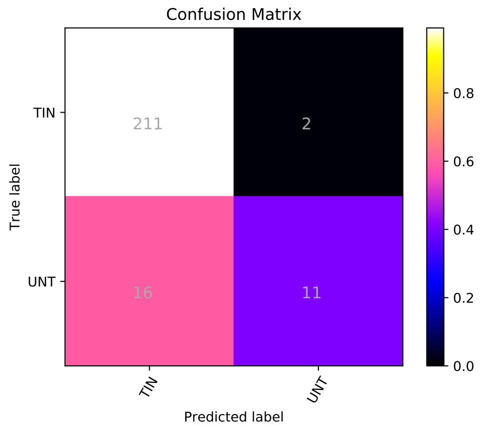


Figure 4: Sub-task B,MSOC method

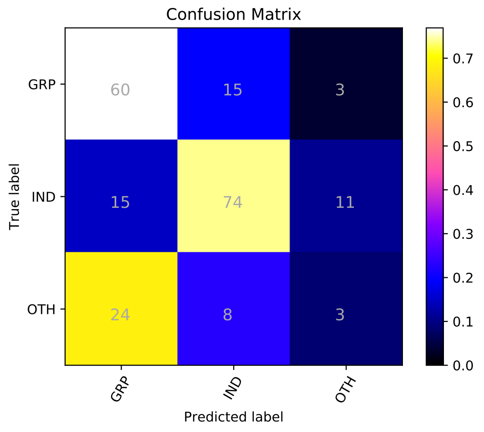


Figure 6: Sub-task C,MSOC method

4.3 Sub-task C - Offense target identification

In offense target identification, RNN method, although has the similar accuracy and F1 score with MSOC method (see Table 4), fails to classify any of the test sentences into 'OTH' class.(see Figure 5) The main reason of this result is 'OTH' class is not as characteristic as other two classes and the partition of this class is the smallest as well. On contrast, MSOC method can successfully classify some test sentences in 'OTH' class. (see Figure 6) This may contribute to the predefined dictionary and sentence structure.

System	F1 (macro)	Accuracy
All GRP	0.1787	0.3662
All IND	0.2130	0.4695
All OTH	0.0941	0.1643
MSOC	0.5149	0.6432
RNN	0.4630	0.6432

Table 4: Results for Sub-task C.

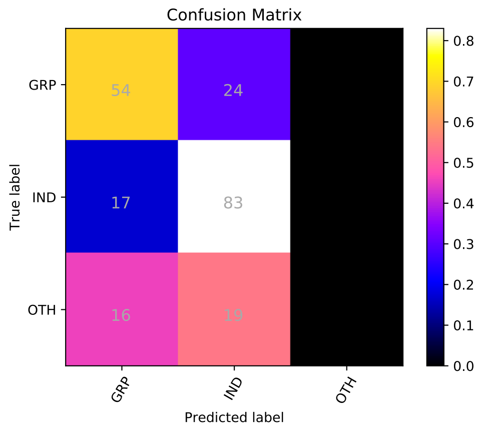


Figure 5: Sub-task C,RNN method

5 Conclusion

RNN is an easy-implemented and high-efficiency method to solve classification problem in natural language processing. In this case, RNN shows an acceptable result but it has many obvious drawbacks. Such as high recall rate when handling unbalanced data, fail to classify certain class if the class is lack of obvious character. The MSOC mehod, on the contrary, can give classification result of same quality. Even though MSOC cannot improve the accuracy or the F1 score of classification to a great extend, we believe we can combine this method with deep learning method to get a better result in similar problems in the future.

References

Y. Chen, Y. Zhou, S. Zhu, and H. Xu. 2012. Detecting offensive language in social media to protect adolescent online safety. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 71–80.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *CoRR*, abs/1605.05101.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*.

Leandro Araújo Silva, Mainack Mondal, Denzil Correa, Fabrício Benevenuto, and Ingmar Weber. 2016. Analyzing the targets of hate in online social media. *CoRR*, abs/1603.07709.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Changli Zhang, Daniel Zeng, Jiexun Li, Fei-Yue Wang, and Wanli Zuo. 2009. Sentiment analysis of chinese documents: From sentence to document level. *Journal of the American Society for Information Science and Technology*, 60(12):2474–2487.

JTML at SemEval-2019 Task 6: Offensive Tweets Identification using Convolutional Neural Networks

Johnny Torres
ESPOL University
jomatorr@espol.edu.ec

Carmen Vaca
ESPOL University
cvaca@fiec.espol.edu.ec

Abstract

In this paper, we propose the use of a Convolutional Neural Network (CNN) to identify offensive tweets. We use an end-to-end model (i.e., no preprocessing) and fine-tune pre-trained embeddings (FastText) during training for learning words' representation. We compare the proposed CNN model to a baseline model, such as Linear Regression, and several neural models. The results show that CNN outperforms other models, and stands as a simple but strong baseline in comparison to other systems submitted to the Shared Task.

1 Introduction

The fast growth of online social networks (OSNs) has provided a medium for users to express their ideas and opinions about any topic. However, some users post offensive content which may deter other users from engaging in online discussions. Despite the tools provided by some OSNs to *block* other users and *report* offensive content, the manual verification of these events are limited in scale and costs due to a large number of malicious events performed by users or bots. Therefore, it is critical to developing automated tools to moderate the content that are robust to ambiguity, sarcasm, and adversarial attacks (Fortuna and Nunes, 2018).

Offensive language detection is an active research area, and several research efforts aim to contribute datasets, propose taxonomies, and improve current models to identify offensive content. In this direction, Zampieri et al. (2019b) proposed a shared task for Identifying and Categorizing Offensive Language in Social Media. The shared task is composed of the following subtasks: **a)** Offensive language identification, **b)** Automatic categorization of offense types, and **c)** Offense target identification.

tweet	Subtask		
	A	B	C
If the tournament of shit ain't on here...	OFF	UNT	-
He is so full of BS!	OFF	TIN	IND
swear niggas make me wanna turn this phone off	OFF	TIN	GRP
Kick the absolute shite out of the car.	OFF	TIN	OTH

Table 1: Examples of Offensive Tweets in the dataset.

Table 1 shows some examples in the dataset of the shared task, and the labels in each of the subtasks. The labels indicate if the tweet is Offensive (OFF) and if it is an untargeted (UNT) or targeted (TIN) offense. The targets of the offensive tweets are individual (IND), group (GRP), other (OTH). This paper contributes specifically to the subtask A in the shared task.

The unstructured and noisy nature of user-generated content on OSNs poses a challenge for classification models. Traditional approaches use a sparse representation for text data, such as the bag of words (BOW) or TF-IDF (Manning et al., 2008).

We propose a model based on Convolutional Neural Networks (CNN) to identify and categorize offensive language on tweets. The learning representation relies on FastText pre-trained word embeddings (Mikolov et al., 2018). Although, this paper focus only on the first subtask, it can be extended to learn the other subtasks.

The rest of the paper describes related work in section 2. Then, we explain in detail our proposed model in section 3, and section 4 shows the results. Finally, we outline the conclusions and future work in section 5.

2 Related Work

Previous work has studied several types of online misbehavior such as aggression (Cheng et al., 2015), cyberbullying (Pieschl et al., 2015), hate speech (Saleem et al., 2017), offensive, and abusive language identification (Waseem et al., 2017).

The major challenge in studying online misbehavior is the several forms it can take and the lack of a standard definition (Saleem et al., 2017). (Waseem et al., 2017) proposed a typology of abusive language sub-tasks. Similarly, a taxonomy proposed to detect toxic messages on Wikipedia discussion pages demonstrated the impact on community health both on and offline (Wulczyn et al., 2017). Wikimedia Foundation found that 54% of contributors decreased participation in the activities when they suffer harassment [1]. The same impact could happen on social media when aggression and offensive language deter other users from engaging in online discussions.

Previous works introduced several datasets like the Internet Argument Corpus (Walker et al., 2012) and the "Hate Speech Twitter Annotations" corpus (Waseem and Hovy, 2016). Most datasets are small in comparison to the Wikipedia dataset (Wulczyn et al., 2017), which enables to train neural models on a large-scale dataset.

In the multilingual aspect, several studies tackle languages other than English like Chinese (Su et al., 2017), Slovene (Fišer et al., 2017), and related shared tasks such as GermEval (Wiegand et al., 2018). However, the studies tackle each language individually due to the difficulty for automated systems to handle multiple languages as idiomatic expressions are dependent on the location and culture. Recent research on identifying profanity vs. hate speech highlighted the challenges of distinguishing between profanity and threatening language which may not contain profane language (Malmasi and Zampieri, 2018).

Recent surveys by (Schmidt and Wiegand, 2017) and (Fortuna and Nunes, 2018) summarizes the taxonomies and methods proposed for detecting abusive language. Also, recent work by (Davidson et al., 2017) introduces the Hate Speech Detection dataset used in several studies (Malmasi and Zampieri, 2017; ElSherief et al., 2018; Zhang et al., 2018).

[1] https://upload.wikimedia.org/wikipedia/commons/5/52/Harassment_Survey_2015_-_Results_Report.pdf

3 Methodology

The model architecture, shown in Figure 1, is a slight variant of the CNN architecture proposed by Kim (2014). We define $\mathbf{x}_i \in \mathbb{R}^k$ as the k-dimensional word vector (i.e., word embeddings) corresponding to the i-th word in the tweets. We padded the tweets to make all equal length, and represent a tweet of length n as

$$\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \ldots \oplus \mathbf{x}_n, \tag{1}$$

where $\oplus$ is the concatenation operator. In general, we refer $\mathbf{x}_{i:i+j}$ to the concatenation of words $\mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_{i+j}$. Then, we apply a convolution operation that uses a *filter* $\mathbf{w} \in \mathbb{R}^{hk}$, over a window of h words to produce a new feature. For example, we generate feature c_i from a window of words $\mathbf{x}_{i:i+h-1}$ by

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b). \tag{2}$$

We denote $b \in \mathbb{R}$ as the bias term and f as a RELU activation function defined as $f(x) = x^+ = \max(0, x)$, where x in the input to the neuron (Glorot et al., 2011) . The convolution layer applies the filter to each possible window of words in the sentence $\{\mathbf{x}_{1:h}, \mathbf{x}_{2:h+1}, \ldots, \mathbf{x}_{n-h+1:n}\}$ to produce a *feature map*

$$\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}], \tag{3}$$

with $\mathbf{c} \in \mathbb{R}^{n-h+1}$. Then, we apply a max-over-time pooling operation over the feature map and take the maximum value $\hat{c} = \max\{\mathbf{c}\}$ as the feature corresponding to this particular filter. The goal is to capture the essential feature (the highest feature value) for the feature maps. The pooling scheme allows us to deal with variable sentence lengths.

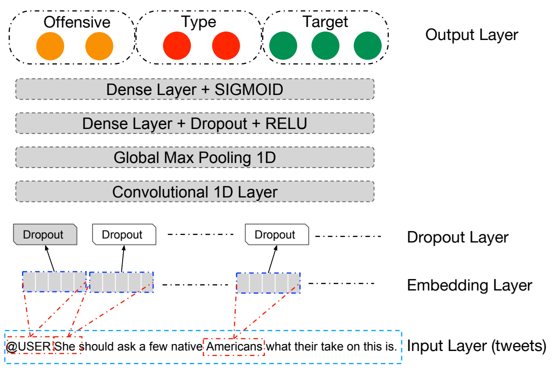


Figure 1: The CNN architecture used to identify offensive tweets using binary output layer.

We have described the process by which we extract *one* feature from *one* filter. The model uses multiple filters to obtain multiple features. These features feed a fully connected layer with a RELU activation function, and finally a sigmoid layer that outputs the probability distribution over labels.

For regularization, we employ a dropout layer with rate $r = 0.2$, constrained on l_2-norms of the weight vectors. We apply the dropout after the embeddings and the penultimate layer. Dropout prevents co-adaptation of hidden units by randomly dropping out (i.e., set to zero) a proportion of p of the hidden units during forward-backpropagation. That is, given the penultimate layer $\mathbf{z} = [\hat{c}_1, \ldots, \hat{c}_m]$ (note that here we have m filters), instead of using

$$y = \mathbf{w} \cdot \mathbf{z} + b \tag{4}$$

for output unit y in forward propagation, dropout uses

$$y = \mathbf{w} \cdot (\mathbf{z} \circ \mathbf{r}) + b, \tag{5}$$

where $\circ$ is the element-wise multiplication operator and $\mathbf{r} \in \mathbb{R}^m$ is a *masking* vector of Bernoulli random variables with probability p of being 1. Gradients are backpropagated only through the unmasked units. At test time, the learned weight vectors are scaled by p such that $\hat{\mathbf{w}} = p\mathbf{w}$, and $\hat{\mathbf{w}}$ is used (without dropout) to score unseen sentences. We additionally constrain l_2-norms of the weight vectors by rescaling $\mathbf{w}$ to have $||\mathbf{w}||_2 = s$ whenever $||\mathbf{w}||_2 > s$ after a gradient descent step.

4 Experiments

In this section, we describe the experimental settings and the results for the subtask A: identifying offensive tweets. We use the data provided in the shared task OffensEval described in Zampieri et al. (2019a). Table 2 describe the label distribution for each of the subtasks. We use the F_1 score as an evaluation metric for the models, and it is the official ranking metric for the shared task is macro-averaged F1.

Subtask A	tweets
NOT	8840
OFF	4400

Table 2: Distribution of the labels in the training dataset.

4.1 Embeddings

We evaluate the CNN model with several word embeddings such as: **a)** Random Uniform initialized, **b)** Word2Vec (Mikolov et al., 2013), and **c)** FastText (Mikolov et al., 2018).

During training, we fine-tune the embedding layer for each type of embeddings. Table 3 shows that FastText embeddings provide the best results, and we use it in further experiments.

Embedding	**Precision**	**Recall**	F_1
Random	71.69	69.47	70.23
Word2Vec	70.67	70.15	70.38
FastText	**71.76**	**71.97**	**71.86**

Table 3: Evaluation of different embeddings.

4.2 Models

We compare the CNN model against baseline models such.

Logistic Regression (LR) with *liblinear* solver and *class weight* to account for the imbalance of the labels.

FastText as a simple and efficient baseline for text classification, and often on par with deep learning classifiers regarding the accuracy but orders of magnitude faster for training and evaluation (Joulin et al., 2016).

LSTM in its vanilla implementation (Tang et al., 2015), with one LSTM layer after the Embeddings Layer.

Bi-LSTM implements a bi-directional LSTM architecture (Zhou et al., 2016).

Table 4 shows the performance of cross-validation data for the proposed CNN model and the baseline models. For evaluation purposes, we split the training dataset in 80% for training subset and 20% testing subset. We use k fold cross validation ($k = 10$) on the training subset. The CNN model outperforms other models in detecting Offensive Tweets and the overall Macro F_1 but detecting Not Offensive tweets works better with Bi-LSTM. We found that the non-neural LR model outperforms neural models such as FastText and LSTM. Bi-LSTM and CNN performance

	Not Offensive			Offensive			Macro		
Model	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
LR	81.80	75.90	78.74	58.27	66.59	62.15	70.03	71.24	70.45
CNN	81.33	80.50	80.91	62.18	63.44	**62.81**	71.76	71.97	**71.86**
LSTM	77.73	78.97	78.34	57.03	55.23	56.11	67.38	67.10	67.23
Bi-LSTM	80.68	81.92	**81.30**	63.11	61.19	62.14	71.90	71.56	71.72
FastText	77.87	79.02	78.44	57.24	55.57	56.39	67.56	67.30	67.42

Table 4: Benchmark of supervised learning models. CNN yields the best performance based on the metric F_1.

are on pair, and further evaluation of the hyper-parameters (e.g., number of layers/neurons, activation functions) is required to determine which of them performs better.

Table 5 show the results in the shared task evaluated on the testing dataset for subtask A. We include a random baseline generated by assigning the same labels for all instances. For example, ”All OFF” in sub-task A represents the performance of a system that labels everything as offensive. The CNN model outperforms by a large margin the random baseline.

System	F_1 (macro)	Accuracy
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
CNN	**0.7591**	**0.8105**

Table 5: Results for Sub-task A using CNN model compared to simple baseline.

Figure 2 shows the confusion matrix for the results with our CNN model. Due to the imbalance in the labels, the False Negatives in the results affects by a large margin the F_1 macro score.

5 Conclusion

In this paper, we proposed a neural model based on Convolutional Neural Networks to identify and categorize offensive tweets on social media. The model outperforms baseline models and other Sequential Models such as LSTM and Bi-LSTM. The reason CNN perform better than sequential models could be due to the noisy and unstructured form of the tweets.

In future work, we plan to use several variations of CNN such as multi-channel and multi-view architectures. Also, we will use recent advances in learning representations based on deep contextualized embeddings such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018).

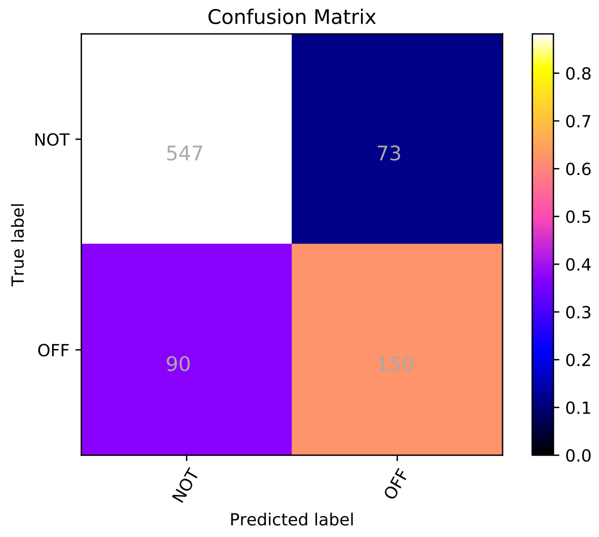


Figure 2: Confusion matrix for Sub-task A, JTML CodaLab CNN model.

References

Justin Cheng, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. 2015. Antisocial behavior in online discussion communities. In *Icwsm*, pages 61–70.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Darja Fišer, Tomaž Erjavec, and Nikola Ljubešić. 2017. Legal Framework, Dataset and Annotation Schema for Socially Unacceptable On-line Discourse Practices in Slovene. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 467–472.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. Scoring, term weighting and the vector space model. *Introduction to information retrieval*, 100:2–4.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Stephanie Pieschl, Christina Kuhlmann, and Torsten Porsch. 2015. Beware of publicity! perceived distress of negative cyber incidents and implications for defining cyberbullying. *Journal of School Violence*, 14(1):111–132.

Haji Mohammad Saleem, Kelly P. Dillon, Susan Benesch, and Derek Ruths. 2017. A web of hate: Tackling hateful speech in online social spaces. *CoRR*, abs/1709.10159.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Huei-Po Su, Chen-Jie Huang, Hao-Tsung Chang, and Chuan-Jie Lin. 2017. Rephrasing Profanity in Chinese Text. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432.

Marilyn A Walker, Jean E Fox Tree, Pranav Anand, Rob Abbott, and Joseph King. 2012. A corpus for research on deliberation and debate. In *LREC*, pages 812–817.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Language Online*.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval*.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399. International World Wide Web Conferences Steering Committee.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3485–3495.

JU_ETCE_17_21 at SemEval-2019 Task 6: Efficient Machine Learning and Neural Network Approaches for Identifying and Categorizing Offensive Language in Tweets

Mainak Pal[*], **Preeti Mukherjee**[*], **Somnath Banerjee, Sudip Kumar Naskar**
Jadavpur University, Kolkata, India
{mainak.pal08,preetimukherjee08,sb.cse.ju}@gmail.com
sudip.naskar@cse.jdvu.ac.in

Abstract

This paper describes our system submissions as part of our participation (team name: JU_ETCE_17_21) in the SemEval 2019 shared task 6: "OffensEval: Identifying and Categorizing Offensive Language in Social Media". We participated in all the three sub-tasks: i) Sub-task A: offensive language identification, ii) Sub-task B: automatic categorization of offense types, and iii) Sub-task C: offense target identification. We employed machine learning as well as deep learning approaches for the sub-tasks. We employed Convolutional Neural Network (CNN) and Recursive Neural Network (RNN) Long Short-Term Memory (LSTM) with pre-trained word embeddings. We used both word2vec and Glove pre-trained word embeddings. We obtained the best F1-score using CNN based model for sub-task A, LSTM based model for sub-task B and Logistic Regression based model for sub-task C. Our best submissions achieved 0.7844, 0.5459 and 0.48 F1-scores for sub-task A, sub-task B and sub-task C respectively.

1 Introduction

Today, very large amounts of information are available in online documents. As part of the effort to better organize this information for users, researchers have been actively investigating the problem of automatic text categorization. Tweets are short length pieces of text, usually written in informal style that contain abbreviations, misspellings and creative syntax (like emoticons, hashtags etc). In this paper we show that our multi-view ensemble approach that leverages simple representations of texts may achieve good results in the task of message polarity classification. We used different machine learning algorithm and neural network approaches for all the tasks which are explained in the subsequent sections. The paper is organized as follows: Section 2 lists down the related work and Section 3 describes our approach. Section 4 presents the experiments, results on the development set and discussion about the confusion matrix and Section 5 details about the observation. Section 6 concludes the paper with possible future work.

OffensEval@SemEval-2019 shared task description, data and results are described in the overview paper (Zampieri et al., 2019b).

[*]These two authors have contributed equally

2 Related Work

Papers published in the last two years include the surveys by (Schmidt and Wiegand, 2017) and (Fortuna and Nunes, 2018), the paper by (Davidson et al., 2017) presenting the Hate Speech Detection dataset used in (Malmasi and Zampieri, 2017) and a few other recent papers such as (ElSherief et al., 2018; Gambäck and Sikdar, 2017; Zhang et al., 2018).

A proposal of typology of abusive language sub-tasks is presented in (Waseem et al., 2017). For studies on languages other than English see (Su et al., 2017) on Chinese and (Fišer et al., 2017) on Slovene. Finally, for recent discussion on identifying profanity vs. hate speech see (Malmasi and Zampieri, 2018). This work highlighted the challenges of distinguishing between profanity, and threatening language which may not actually contain profane language.

In addition, we would also like to mention the previous editions of related workshops such as TA-COS[1], Abusive Language Online[2], and TRAC[3] and related shared tasks such as GermEval (Wiegand et al., 2018) and TRAC (Kumar et al., 2018).

3 Methodology and Data

3.1 Data Description

The organizers provided a dataset of 13,240 tweets which were annotated with the following task-specific categories.

- **Sub-task A:** Offensive language identification.
  1. Not Offensive (*NOT*): These posts do not contain offense or profanity.
  2. Offensive (*OFF*): These posts contain offensive language or a targeted (veiled or direct) offense.
- **Sub-task B:** Automatic categorization of offense types.
  1. Targeted Insult and Threats (*TIN*): A post containing an insult or threat to an individual, a group, or others (see categories in sub-task C).
  2. Untargeted (*UNT*): A post containing non-targeted profanity and swearing.
- **Sub-task C:** Offense target identification.
  1. Individual (*IND*): The target of the offensive post is an individual: a famous person, a named individual or an unnamed person interacting in the conversation.
  2. Group (*GRP*): The target of the offensive post is a group of people considered as a unity due to the same ethnicity, gender or sexual orientation, political affiliation, religious belief, or something else.
  3. Other (*OTH*): The target of the offensive post does not belong to any of the previous two categories (e.g., an organization, a situation, an event, or an issue).

[1] http://ta-cos.org/
[2] https://sites.google.com/site/abusivelanguageworkshop2017/
[3] https://sites.google.com/view/trac1/home

Table 1: Statistics of the training dataset

NOT				8040
OFF	UNT			524
	TIN	IND	2407	3876
		GRP	1074	
		OTH	395	
TOTAL				12440

The data collection methods used to compile the dataset used in OffensEval is described in Zampieri et al. (2019a). Table 1 provides statistics of the training dataset.

3.2 Preprocessing

Raw tweets scraped from twitter generally result in a noisy dataset. This is due to the casual nature of people's usage of social media. Tweets have certain special characteristics such as re-tweets, emoticons, user mention, etc. which have to be suitably extracted. Therefore, raw twitter data has to be normalized to create a dataset which can be easily learned by various classifiers. We applied an extensive number of pre-processing steps to standardize the dataset and reduce its size. Initially, we performed basic pre-processing operations on tweets which are as follows:

1. Convert the tweets to lower case.
2. Selective removal of special twitter features like URL, User mention, Hash-tags etc. (Cf. Table 2)
3. Converting abbreviated negative english words to common negative verbs.
4. Removing special characters and numbers.
5. Tokenization.

Table 2: Regex used for pre-processing

Twitter Feature	Regex pattern
URL	https?://[^]+ | www.[^]+
Mention	@[A-Za-z0-9]+
Hashtags	#[A-Za-z0-9]+

3.3 Machine Learning

Most of the machine learning (ML) algorithms are heavily reliant on hand crafted features designed by experts. This makes ML algorithms less generalizable. So we did not use any language specific features. We used various Machine Learning techniques to classify the tweets. When comparing various machine learning algorithms, baseline provides a point of reference to compare. While developing the models, we employed TextBlob[4] as baseline. We compared the validation result with TextBlob. Textblob is a python library for processing textual data. Apart from useful tools such as POS tagging, n-gram,etc. it has a built-in sentiment classification tool. We also tried a variation for the fine-grained classification task where the predicted output from task A was also added as a feature to the TF-IDF and list specific features. We validated our models using 15% of the training data. We built an ensemble (voting) classifier with top 5 models for different types of vectorizers, number of features, n-grams, etc.

3.4 Convolutional Neural Network

Word embedding: We used Glove[5] as the vector representation of the words in tweets. The dimension of the embedding is 300. We fine-tuned the word embedding during the network training.

Network Architecture: As shown in the Figure 1 embedding layer is used to provide word embedding. We used 300 dimensional word vectors for each words. We used 1D CNN on text data represented in word vectors. Filter column width is same as the data column width. It will ensure that matrix will stride vertically only. The padded data of the input text is of size 65x300 for each sentences. Therefore, filter's column width will be 300. Height is similar to the concept of n-gram. If the filter height is 2, the filter will stride through the document computing the calculations with all the bigrams; if the filter height is 3, it will go through all the trigrams in the document, and so on. The output height is measured by the following mathematical expression :

$$Output\ height = ((H - hf)/s) + 1$$

where, H: Input data height hf: Filter height s: Stride size

[4]https://textblob.readthedocs.io/en/dev/
[5]https://nlp.stanford.edu/projects/glove/

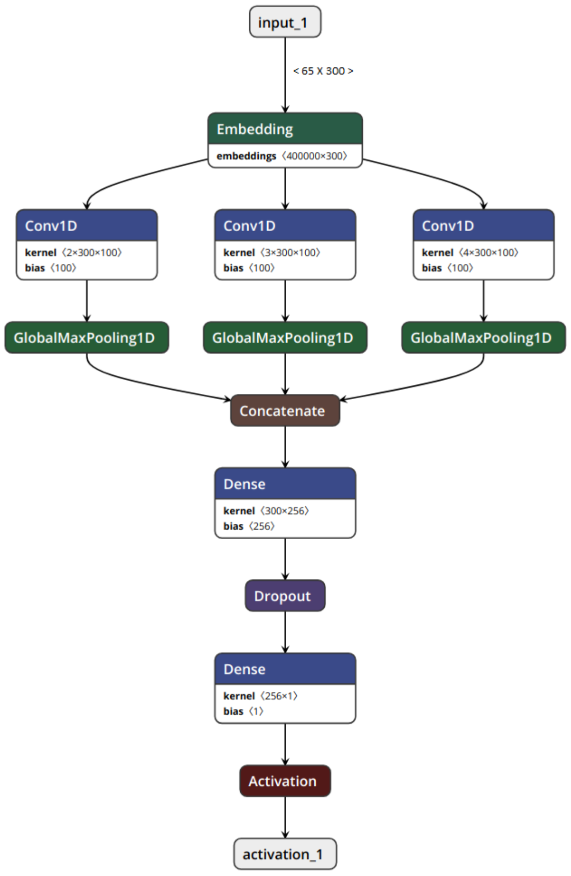


Figure 1: Convolutional Neural Network

Global Max Pooling layer extracts the maximum value from each filter, and the output dimension is 1-dimensional vector with length as same as the number of filters we applied. This can be directly passed on to a dense layer without flattening.

We implemented the above with bi-gram, tri-gram and four-gram filters. However, this is not linearly stacked layers, but parallel layers. And after convolutional layer and max-pooling layer, it simply concatenated max pooled result from each of bi-gram, tri-gram, and four-gram, then build one output layer on top of them. We added one fully connected hidden layer with dropout just before the output layer. Output layer has just one output node with Sigmoid activation.

3.5 Recurrent Neural Networks

Long Short-Term Memory networks are an extension for RNN. We employed LSTM as RNN architecture.

Word embedding: Here, we also used Glove as the vector representation of the words in tweets. The dimension of the embedding is 200. We fine-tuned the word embedding during the network training.

Network Architecture: The matrix contains 400,000 word vectors, each with a dimensionality mentioned above. We imported two different data structures, one was a Python list with the 400,000 words, and another was a $400,000 \times 200$ dimensional embedding matrix that holds all of the word vector values. We defined the necessary hyper-parameters and specified the two placeholders, one for the inputs into the network, and one for the labels. The most important part about defining these placeholders was understanding each of their dimensionality. For both tasks, the output layer contained nodes equal to the number of class labels(2 for task A and B, 3 for task C).

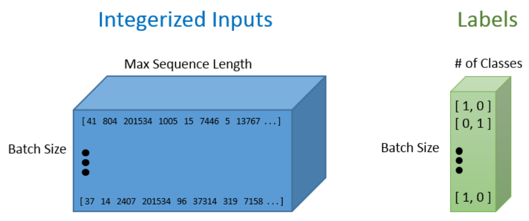


Figure 2: Vectorized tweets and corresponding labels

Each row in the integerized input placeholder represents the integerized representation of each training example that we included in our batch. Hidden state vector can be represented as :

$$h_t = \sigma(w^H h_{t-1} + w^X x_t)$$

where, w^H and w^X are weight metrics, x_t is a vector that encapsulates all the information of a specific word.

We also used LSTM network as a module in RNN for better understanding of a sentence. All the vectors are given as a sequence of vectors for a bidirectional LSTM. The representation of a tweet is the representation learned after passing the whole sequence of tokens through the biLSTM. We defined a standard cross entropy loss with a softmax layer put on top of the final prediction values. For the optimizer, we used Adam and the default learning rate of 0.001.

4 Results

This section presents the obtained results for the three sub-tasks.

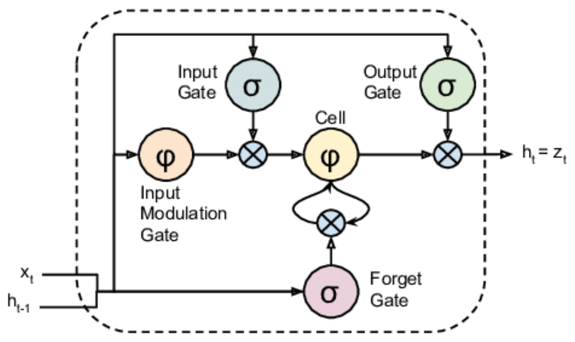


Figure 3: LSTM unit

4.1 Sub-task A:

We implemented all the three systems for this sub-task. For Machine learning, we obtained best results for Count-vectorizer with tri-gram and 90,000 features and with Logistic Regression Classifier. We achieved best results for CNN-Glove with Macro-F1 0.7844 and overall Accuracy 0.8419. However, due to paucity of time, we were unable to extract the output from our RNN model in the stipulated time frame.

System	F1 (macro)	Accuracy
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
ML model	0.7231	0.8105
CNN-glove	**0.7844**	**0.8419**

Table 3: Results for Sub-task A.

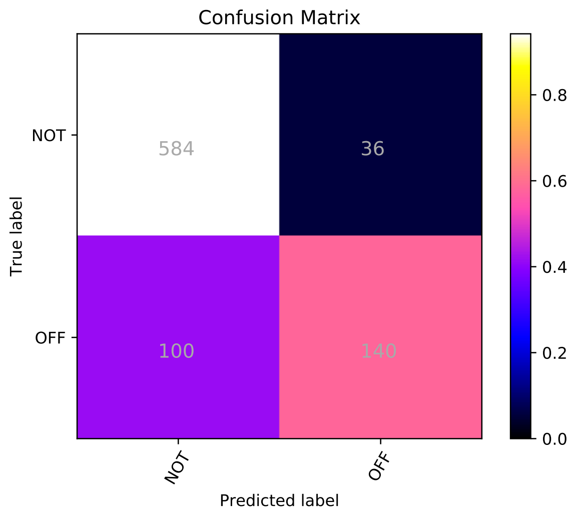


Figure 4: Confusion matrix of CNN-glove model for Sub-task A

4.2 Sub-task B:

Our approach was similar to that in the previous Sub-task. We changed the training set and labels of the same appropriately, and got our results. We used 2 class layers for training.We observed that the model gives better validation accuracy while fitted with cleaned data parsing with @*user*. While training the RNN network, we used alternative targeted and non-targeted tweets from annotated data. We obtained best results for RNN with Macro-F1 0.54587543782 and overall Accuracy 0.804166666667. The Hyper-parameters of this model are: Batchsize:24, LSTM Units:64, Epochs Number:1,00,000, Glove embeddings:200D, Optimizer:Adam. In Machine Learning, we used several traditional techniques. Best validation accuracy was found for Logistic Regression as classifier, countvectorizer - trigram - 50k feature.

System	F1 (macro)	Accuracy
All TIN baseline	0.4702	0.8875
All UNT baseline	0.1011	0.1125
ML model	0.5378	**0.8917**
RNN-LSTM	**0.5459**	0.8042

Table 4: Results for Sub-task B.

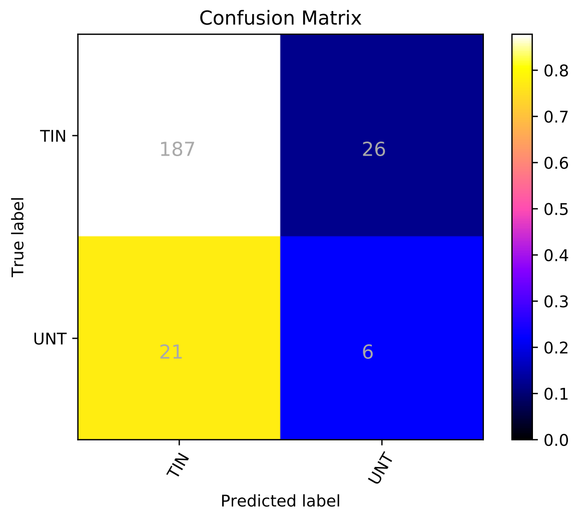


Figure 5: Confusion matrix of RNN-LSTM model for Sub-task B

4.3 Sub-task C:

This task was the most challenging among the three tasks because of the small training data. The training data contains only 3876 tweets and the 3 sub-classes are unevenly distributed. First of all, since this was a ternary classification task, we could only pursue a handful of machine learning algorithms and secondly for neural network architectures, there is a paucity of huge dataset to train the model properly. We used 10% of training dataset for testing and validation purposes, while the rest used for training. We converted our text documents to a matrix of token counts (CountVectorizer), then transformed a count matrix to a normalized tf-idf representation (tf-idf transformer). After that, we trained several classifiers from Scikit-Learn[6] library. Now among the various classifiers, we built an ensemble (voting) classifier with top 5 models and found the best accuracy result for Logistic Regression. To make the vectorizer transformer-classifier easier to work with, we used Pipeline class in Scikit-Learn that behaves like a compound classifier. For RNN, the same previous system was used but with some alterations as change in labels and change in iterative conditions for output prediction as this was a ternary classification task. We obtained best results for Machine Learning with Logistic Regression Classifier with 0.480057590252,0.577464788732 in terms of Macro-F1 and overall Accuracy respectively.

System	F1 (macro)	Accuracy
All GRP baseline	0.1787	0.3662
All IND baseline	0.2130	0.4695
All OTH baseline	0.0941	0.1643
RNN-LSTM	0.4580	0.5681
CNN-glove	0.4352	**0.6056**
ML Model	**0.4801**	0.5775

Table 5: Results for Sub-task C.

5 Observations

We noticed that both the F1(macro) and accuracy are high, in Sub-task A. This is probably due to relatively large size of training data. In sub-task B, we have found that, though the accuracy is optimum, F1(macro) is surprisingly low. This is due to imbalanced dataset. Many classes have fewer samples to create robust models. This goes same for the sub-task C .

[6]https://scikit-learn.org/stable/

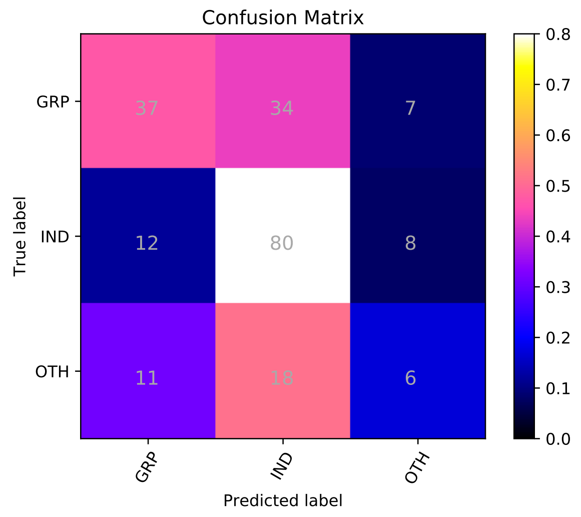


Figure 6: Confusion matrix of ML model for Sub-task C

6 Conclusions

In this paper, we have briefly described our submissions to SemEval2019 Task 6 on Identification and Categorization of Offensive Language on Twitter data. Our systems ranked 21st out of 103 participants for Sub-task A, 50th out of 75 participants for Sub-task B and 47th out of 66 participants for Sub-task C.Although our validation accuracy was high, the F1-score primarily dropped due to unequal distribution of opposite polarity data.

We could have made the system work better by training our model with additional tweets which we could have annotated manually. We could have also used Siamese Network to train our model, which has been generally used for image data.

References

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Darja Fišer, Tomaž Erjavec, and Nikola Ljubešić. 2017. Legal Framework, Dataset and Annotation Schema for Socially Unacceptable On-line Discourse Practices in Slovene. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 467–472.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Huei-Po Su, Chen-Jie Huang, Hao-Tsung Chang, and Chuan-Jie Lin. 2017. Rephrasing Profanity in Chinese Text. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

KMI−Coling at SemEval-2019 Task 6: Exploring N-grams for Offensive Language detection

Priya Rani
Dr. Bhimrao Ambedkar University
Agra, India
pranijnu@gmail.com

Atul Kr. Ojha
Jawaharlal Nehru University
New Delhi, India
shashwatup9k@gmail.com

Abstract

In this paper, we present the system description of offensive language detection tool which is developed by the KMI−Coling Group under the OffensEval Shared task. The OffensEval Shared Task was conducted in SemEval 2019 workshop. To develop the system, we have explored n-grams up to 8-gram and trained three different systems namely A, B and C system for three different sub tasks within the OffensEval task which achieves the accuracy of 79.76%, 87.91% and 44.37% respectively. The task was completed using the data set provided to us by OffensEval organisers, which was the part of OLID data set. It consists of 13,240 tweets extracted from twitter and were annotated at three levels using crowd sourcing.

1 Introduction

The very first question which arises in one's mind when one starts working in the area of computational sociolinguistics research related to the language usage in social media and networking sites is what is offensive language and its related terms such as hate speech, aggression and all? The second question arises is related to the definition of this terminology. We would suggest that offensive language is still not a very well-defined phenomenon. As we know that the natural language is productive in nature. These aspect of the language use has always existed as part of the speech repertoire of the speaker. In the area of scientific study, we need to move forward with a definition. Therefore we will go by the definition given by Jay and Janschwetiz which states that Offensive language is vulgar, pornography and hateful language (Chen et al., 2012). But even this definition does not incorporate the many more structure which is neither vulgar nor pornography nor hateful but are definitely offensive. Such type of structure is what leads to challenges in the detection of offensive language in the discourse. With the increase in the culture of social media and social networking sites, the use of offensive language has increased rapidly. Moreover, it has also given a very good platform to conduct different research in the given area.

2 Literature review

This section gives a brief outline of the existing literature and approaches that are available for offensive language detection. Lots of research works are being done to detect offensive language and there has been significant progress over time. Lexical Syntactic based framework was used for sentence offensive detection and user offensive detection by Chen et al. (2012). Another study by Xiang et al. (2012) which uses keyword matching technique that performed very well in literature domain. Razavi et al. (2010) uses auxiliary weighted repository by matching the text to its graded entries with the help of both rule-based and statistical pattern to detect flames from the text. Maisto et al. (2017) uses a lexicon-based method for the automatic identification and classification.

3 System Overview

We built three different systems for three sub tasks in the shared task. The system was built using a supervised machine learning approach trained on different classifiers using n-gram model.

3.1 System A

The very first was developed to detect whether the tweets are offensive or not. The system uses unigram and bigram in the feature set and was trained on Linear SVM classifier.

3.2 System B

The second system was developed to detect whether the tweets are targeted or non-targeted only if the tweets are offensive in nature. This system has also been trained on linear SVM with n-gram language model which consisted of unigram, bigram, trigram and 4-gram.

3.3 System C

The third system was one step ahead to detect whether the tweets are targeted to an individual, group or other. In which the third category 'other' includes a wide range of categories such as entity, organisation, place, country and many more. The system is trained on decision tree with n-gram feature starting from uni-grams to 8-grams.

4 Experiments

In this section, we briefly describe the experimental settings which are used to develop offensive language detection tool.

4.1 Data set

The data we used to train and test the system was provided by SemEval shared task 2019 under task 6 called OffensEval (Zampieri et al., 2019a). The data set consists of 13,240 annotated tweets which were extracted from OLID, Offensive Language Identification Dataset (Zampieri et al., 2019a). The data set was further divided into training and testing set in the ratio 80:20. We have used the same data set to train and test all the three systems developed to participate in the sub task of Task 6 in SemEval 2019.

4.2 Annotation

The data was hierarchically annotated using crowd sourcing. The gold labels were assigned by taking inter-annotator agreement into consideration. No correction has been carried out on the crowd sourcing annotations. The tweets were annotated at three levels. Level A differentiates the tweets between offensive and non-offensive. Level B category the offensive tweets in another level that whether the offensive tweets are targeted or non-targeted insults or threat. Level C category further categorise the targets of the insult into three different categories as an individual, group or other (Zampieri et al., 2019a).

4.3 Development of systems for sub tasks

In the next step, we developed three offensive detection systems to detect offensive tweets, targeted insults and to categorise the targeted insults using n-gram language model.

Training and development of system for sub task A
The systems were trained independently on SVM. To explore the role of n-gram feature in the detection of offensive language we have used the scikit-learn toolkit to experiment with unigram, bigram, trigram and 4-gram. We used the tweets and only Label A to train the system for development of system A.

Training and development of system for sub task B
Like system A, system B is trained on SVM with the scikit-learn toolkit to experiment with unigram, bigram, trigram and 4-gram. The system was trained and tested using tweets, Label A and Label B.

Training and development of system for sub task C
The third system was trained on two different classifier SVM and Decision tree with the same scikit-learn toolkit. The feature set for the system consists of n-gram ranging from unigrams to 8-gram.s In order to train the system, we have used the tweets, Label B, and Label C.

5 Detailed Error Reports of the KMI−Coling System

This section presents a detailed study of the result that is achieved by the developed systems.
System A performs well with the only unigram when trained with SVM. We also weighted the feature set with TF-IDF but that turned out to give very disappointing results and thus was discarded from the final feature set. Similarly, bigram, trigram and 4-gram decrease the accuracy of the system. The final trained system gave the precision of 78.72%, recall of 79.77% with an F1 score of 78.58%. The confusion matrix of the system providing the detail error is given in figure 1, which shows that 122 offensive tweets were called as non-offensive. Whereas 568 tweets were recognised correct by the system. On the other hand, 52 non-offensive tweets were labelled as offensive tweets and 118 offensive tweets were marked correctly by the system (Zampieri et al., 2019b).

System B, unlike system A, performed well in terms of accuracy, but the recall of non-targeted offensive tweets is lowered when trigram and 4-gram are implemented. Weighting the feature set using TF-IDF also did not work well as it de-

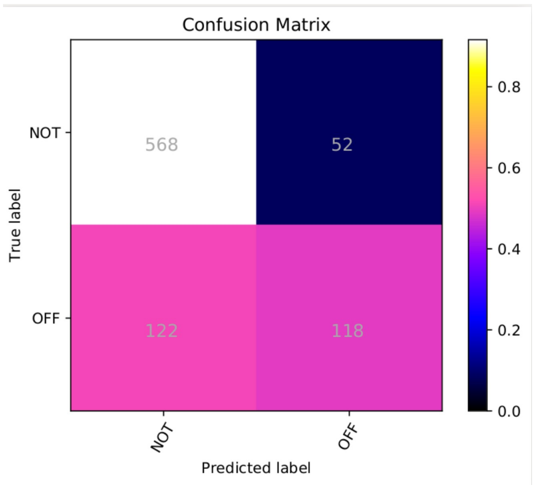


Figure 1: Confusion matrix of sub task A

creases the accuracy of the system. Finally, the system was trained only with unigram and bigram. The overall precision of the system is 84.38%, recall is 87.92% and F1 score is 85.37%. Figure 2 shows the confusion matrix of the system. The matrix gives the error report such that 23 non-targeted tweets were label targeted by the system whereas only 6 targeted tweets were marked as non-targeted.

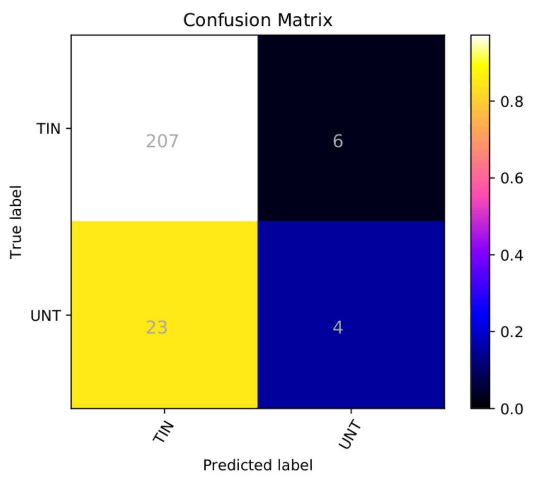


Figure 2: Confusion matrix of sub task B

As we have mentioned above system c was trained on two classifiers SVM and Decision Tree. We will be only reporting the final confusion matrix of the system C which was trained on Decision Tree. The precision of the system is 52.84%, recall is 59.15% and F1 score is 55.31%. We can easily detect and study the error from the confusion matrix given in Figure 3. Twenty tweets which originally belong to other group categorised in GROUP, 14 tweets in INDIVIDUAL. Secondly, 13 tweets which belong to INDIVIDUAL were put into GROUP and 6 of them in OTHER. Thirdly, 30 tweets which were targeted towards a group were labelled in INDIVIDUAL and 4 tweets in OTHER.

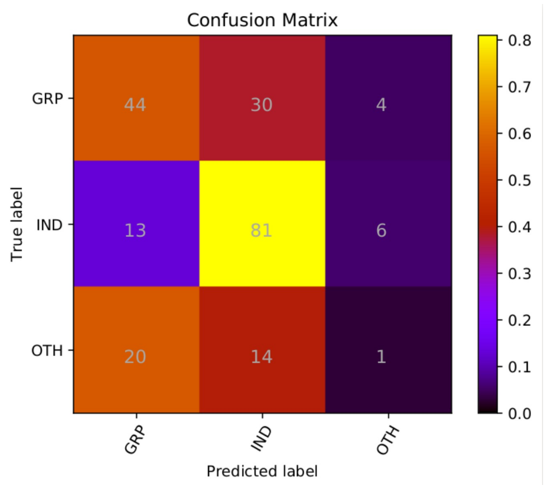


Figure 3: Confusion matrix of sub task C

6 Conclusion

In this paper, we propose an offensive detection tool which is only based on the n-gram model. We have experimented with n-gram model where n = 1,2,3,4,5,6,7,8 via statistical model. The n-gram model has been shown to perform well in very less amount of time in comparison to other models. The accuracy of system A is 79.76%, system B is 87.91% and of system C is 44.37% in our experiment. In addition to this, it is very easy to implement n-gram and consume very less amount of time. Our system can be further improved with the help of neural network. As we can see that the n-gram model also accommodate the phrase level structure from the given text. Therefore, implementing simple sentence feature would not help in increasing the accuracy. The sentence level feature would work only when there is a language specific feature.

Acknowledgments

We are grateful to the organizers of Offsemeval-2019 for providing us with the tweet Corpus and evaluation scores.

References

Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. 2012. Detecting offensive language in social media to protect adolescent online safety. In *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*, pages 71–80. IEEE.

Maral Dadvar, Dolf Trieschnigg, and Franciska de Jong. 2014. Experts and machines against bullies: A hybrid approach to detect cyberbullies. In *Canadian Conference on Artificial Intelligence*, pages 275–281. Springer.

H Vinutha Divyashree and NS Deepashree. 2016. An effective approach for cyberbullying detection and avoidance. *International Journal of Innovative Research in Computer and Communication Engineering*, 14.

Love Engman. 2016. Automatic detection of cyberbullying on social media.

Altaf Mahmud, Kazi Zubair Ahmed, and Mumit Khan. 2008. Detecting flames and insults in text.

Alessandro Maisto, Serena Pelosi, Simonetta Vietri, Pierluigi Vitale, and Via Giovanni Paolo II. 2017. Mining offensive language on social media. In *Proocedings of CLiC-it 2017 4th Italian Conference on Computational Linguistics*.

Georgios K Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*.

Amir H Razavi, Diana Inkpen, Sasha Uritsky, and Stan Matwin. 2010. Offensive language detection using multi-level classification. In *Canadian Conference on Artificial Intelligence*, pages 16–27. Springer.

Caitlin Elizabeth Ring. 2013. Hate speech in social media: An exploration of the problem and its proposed solutions.

Semiu Salawu, Yulan He, and Joanna Lumsden. 2017. Approaches to automated detection of cyberbullying: A survey. *IEEE Transactions on Affective Computing*, (1):1–1.

Joni Salminen, Hind Almerekhi, Milica Milenkovic, Soon-gyo Jung, Jisun An, Haewoon Kwak, and Bernard J Jansen. 2018. Anatomy of online hate: Developing a taxonomy and machine learning models for identifying and classifying hate in online news media. In *ICWSM*, pages 330–339.

Sasha Sax. 2016. Flame wars: Automatic insult detection.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Cynthia Van Hee, Gilles Jacobs, Chris Emmery, Bart Desmet, Els Lefever, Ben Verhoeven, Guy De Pauw, Walter Daelemans, and Véronique Hoste. 2018. Automatic detection of cyberbullying in social media text. *arXiv preprint arXiv:1801.05617*.

Guang Xiang, Bin Fan, Ling Wang, Jason Hong, and Carolyn Rose. 2012. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1980–1984. ACM.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

LaSTUS/TALN at SemEval-2019 Task 6: Identification and Categorization of Offensive Language in Social Media with Attention-based Bi-LSTM model

Lutfiye Seda Mut Altin, Àlex Bravo, Horacio Saggion
Large Scale Text Understanding Systems Lab / TALN Research Group
Department of Information and Communication Technologies (DTIC)
Universitat Pompeu Fabra
Tanger 122, Barcelona (08018), Spain
lutfiyeseda.mut01@estudiant.upf.edu, {alex.bravo, horacio.saggion}@upf.edu

Abstract

This paper describes a bidirectional Long-Short Term Memory network for identifying offensive language in Twitter. Our system has been developed in the context of the SemEval 2019 Task 6 which comprises three different sub-tasks, namely A: Offensive Language Detection, B: Categorization of Offensive Language, C: Offensive Language Target Identification. We used a pre-trained Word Embeddings in tweet data, including information about emojis and hashtags. Our approach achieves good performance in the three sub-tasks.

1 Introduction

As the amount of user generated content in social media is increasing at an exponential pace, detecting offensive language and harmful content automatically in an efficient way is a very important issue for the society. Recent work has shown that offensive language in various forms such as hate speech, cyberbullying, profanity and harassment has negative effects especially in adolescents (Hamm et al., 2015).

The shared task, Categorizing Offensive Language in Social Media (SemEval 2019 - Task 6), focuses on improving identification of offensive language by considering type and target of the offense into account (Zampieri et al., 2019b). The task is composed of three sub-tasks. Sub-task A aims to identify if a given tweet is **offensive** or **not** (annotated as **OFF** or **NOT**). Sub-task B aims to categorize the offense type in offensive tweets into two categories: **targeted** (**TIN**) or **untargeted** (**UNT**) meaning that if a tweet contains an insult or threat to an individual, a group or something else or if a tweet contains non-targeted offense such as general profanity or non-acceptable language. Lastly, Sub-task C aims to identify the **target type** of targeted offensive posts. The target type is supposed to be classified as individual, group or other for the rest (annotated as **IND**, **GRP** or **OTH**). We submitted three different runs for each sub-task.

The training dataset released by the shared task organizers, consists of 14,100 English tweets with one annotation layer per task with a hierarchical annotation scheme where each annotation level is related to an independent sub-task. The methods used to collect this dataset is described in (Zampieri et al. (2019a)). Examples from the dataset with annotations at the end are given below:

> *"@USER That shit weird! Lol OFF (offensive) - -"*
>
> *"@USER @USER You are an embarrassing citizen!! OFF TIN -"*
>
> *"@USER @USER Liberals ruin everything! OFF TIN GRP"*

This paper describes a bidirectional Long Short Term Memory network (biLSTM) model with an Attention layer to identify offensive language in Twitter. The rest of the paper is organized as follows: In section 2, we introduce an overview of the work related to identification of offensive language. In Section 3 we describe our model structure and differences between the different runs submitted for each sub-task. In Section 4 we provide the results and discuss the performance of the system. Finally, in Section 5 we conclude giving an outline for the future work.

2 Related Work

Identification of offensive language in user-generated content can essentially be considered as a classification task. Previous research on the

issue has been carried out with approaches from different perspectives such as abusive language (Waseem et al., 2017) (Chu et al., 2017), hate speech (Davidson et al., 2017) (Schmidt and Wiegand, 2017) (Fortuna and Nunes, 2018) and cyberbullying (Hee et al., 2018).
It has been referred by (Kumar et al., 2018) that for identification of aggression in a more general manner, classifiers such as SVM and logistic regression can equalize the results of neural networks-based systems if the right features are selected. On the other hand, (Zhang et al., 2018) pointed out that a deep neural network model combining convolutional neural network and long short term memory network, performed better than state of the art, including SVM. Furthermore, indicated that automatically selected features performed better than manual features.

Recent research also includes the work of (ElSherief et al., 2018) focusing on the target of the hate speech found that in terms of word characteristics, such as frequency of specific words, differences can be observed between hate to individuals or to groups.

(Gambäck and Sikdar, 2017) investigated classification of different sub-categories of hate speech with different Convolutional Neural Network models founding that word2vec embeddings performed best. Davidson et al. worked on distinguishing hate speech and offensive language by training a multi-class classifier showing that using lexicons is useful in agreement with the previous research (Davidson et al., 2017).

Both for English and other languages similar shared tasks have been organized. At GermEval that aims to identify offensive language in German tweets; popular features were lexicons of offensive words, word embeddings and character n-grams. Between deep learning approaches and traditional supervised classifiers there was not a clear superior system in terms of the scores (Wiegand et al., 2018). EVALITA 2018 Hate Speech Detection Shared Task focused on Italian text on Facebook and Twitter. Best performed system in this shared task utilized polarity and subjectivity lexicon with word embeddings (Caselli et al., 2018).

3 Methodology and Data

This paper describes a neural network based on the model proposed by Zhou et al. (2016) for relation extraction. The model consist of a bidirectional Long Short-Term Memory Networks (biLSTM) model with an Attention layer on top. The model capture the most important semantic information in a tweet, including emojis and hashtags, to face the three sub-tasks. In Figure 1 a simplified schema of our model can be seen. In the following we explain how the model works.

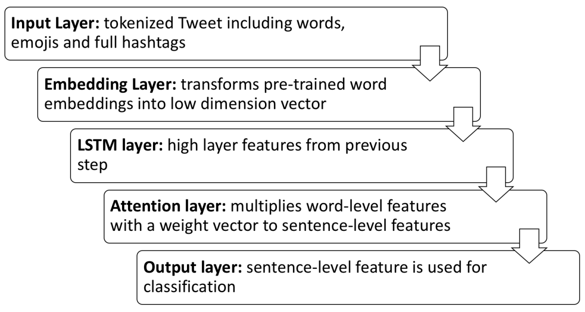

Input Layer: tokenized Tweet including words, emojis and full hashtags
Embedding Layer: transforms pre-trained word embeddings into low dimension vector
LSTM layer: high layer features from previous step
Attention layer: multiplies word-level features with a weight vector to sentence-level features
Output layer: sentence-level feature is used for classification

Figure 1: Simplified schema of the model

First, the tweets were tokenized removing punctuation marks and keeping emojis and full hashtags because can contribute to define the meaning of a tweet.

Second, the embedding layer transforms each element in the tokenized tweet (such as words, emojis and hashtags) into a low-dimension vector. The embedding layer, composed of the vocabulary of the task, was randomly initialized from a uniform distribution (between -0.8 and 0.8 values and with 300 dimensions). Recent studies have reported that pre-trained word embeddings are far more satisfactory than the randomly initialized embeddings (Erhan et al., 2010; Kim, 2014). For that reason, the initialized embedding layer was updated with the word vectors included in a pre-trained model based on all the tokens, emojis and hashtags from 20M English tweets (Barbieri et al., 2016), which were updated during the training.

Then, a biLSTM layer gets high-level features from previous embeddings. The LSTM were introduced by Hochreiter and Schmidhuber (1997) and were explicitly designed to avoid the long-term dependency problem. LSTM systems keep relevant information of inputs by incorporating a loop enabling data to flow from one step to the following. LSTM gets a word embedding sequentially, left to right, at each time step, produces a hidden step and keeps its hidden state through time. Whereas, biLSTM, does the same process as standard LSTM, but processes the text in a left-to-right as well as right-to-left order in parallel. Therefore, gives two hidden state as output at each step and is able to capture backwards and long-range dependencies.

A critical and apparent disadvantage of seq2seq models (such as LSTM) is that they compress all information into a fixed-length vector, causing the incapability of remembering long tweets. Attention mechanism aims to overcome the limitation of fixed-length vector keeping relevant information from long tweet sequences. In addition, attention techniques have been recently demonstrated success in multiple areas of the Natural Language Processing such as question answering, machine translations, speech recognition and relation extraction (Bahdanau et al., 2014; Hermann et al., 2015; Chorowski et al., 2015; Zhou et al., 2016). For that reason, we added an attention layer, which produces a weight vector and merge word-level features from each time step into a tweet-level feature vector, by multiplying the weight vector. Finally, the tweet-level feature vector produced by the previous layers is used for classification task by a fully-connected layer.

Furthermore, we applied dropout regularization in order to alleviate overfitting. Dropout operation sets randomly to zero a proportion of the hidden units during forward propagation, creating more generalizable representations of data. As in Zhou et al. (2016), we employ dropout on the embedding layer, biLSTM layer and before the output layer. The dropout rate was set to 0.5 in all cases.

We used an additional annotated dataset for the sub-task A. This additional dataset was released with Shared Task on Aggression Identification organized as part of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC - 1) (Kumar et al., 2018). This dataset is composed of 15,000 aggression-annotated Facebook Posts that were annotated as Overtly Aggressive, Covertly Aggressive, and Non-aggressive texts. For this sub-task A, posts with aggressive annotations were considered as offensive (OFF) and Non-aggressive annotation as not (NOT).

For every sub-task, three different runs were submitted following the same scheme of the neural network. Specifically, for sub-task A, we submitted 2 runs taking into account the additional dataset (using Adam in the Run1A and RMSProp optimizer in the Run3A). The third run (Run2A) was obtained using only the dataset provided by the organizers and using Adam as optimizer.

For sub-tasks B and C, we did not use additional data for training. Instead, we weighted the classes in the training giving major relevance to unbalanced classes. For the rest of the runs, some parameters were changed in order to obtain different results. Specifically, the Run1B and Run1C the Adam optimizer was used with 50 units in the LSTM. The RMSProp optimizer was used in the Run2B and RUN2C with the previous number of LSTM units. Finally, in the Run3B and Run3C was also applied the RMSProp optimizer but with 100 units in the LSTM. Additionally, to improve the model performance and reducing the overfitting for sub-task C, which contains the smallest number of instances for training, the LSTM layer included a weight regularization (L1 and L2).

4 Results

F1 scores and accuracies of our three different submissions for sub-task A are shown in Table 1. For this sub-task we have achieved the highest score with the system we did not train with an additional dataset. F1 scores and accuracies of all submissions for the subsequent tasks are seen in Table 2 and 3 respectively.

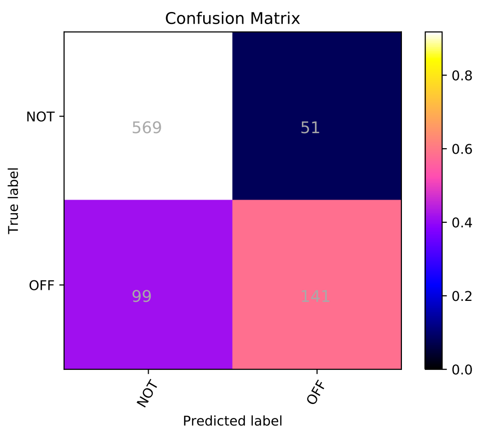


Figure 2: Confusion matrix of our best performed model for Sub-task A (biLSTM with specific configuration - Run2A)

The confusion matrix of our best performed model for the first task (see Figure 2) illustrates that between the two classes, NOT (not offensive) class achieves the best result where the majority of the data being correctly classified.

For sub-task B, classification of TIN (targeted

System	F1 (macro)	Accuracy
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
LaSTUS/TALN - Run1A	0.7406	0.7860
LaSTUS/TALN - Run2A	**0.7682**	**0.8256**
LaSTUS/TALN - Run3A	0.7411	0.7872

Table 1: Results of different submissions for **Sub-task A**.

System	F1 (macro)	Accuracy
All TIN baseline	0.4702	0.8875
All UNT baseline	0.1011	0.1125
LaSTUS/TALN - Run1B	0.6425	0.8458
LaSTUS/TALN - Run2B	0.6150	0.8292
LaSTUS/TALN - Run3B	**0.6618**	**0.8542**

Table 2: Results of different submissions for **Sub-task B**.

insult and threads) and UNT (untargeted) content from a sub-set of offensive tweets, the confusion matrix demonstrates that our best performed model has the highest precision for class TIN as shown in Figure 3.

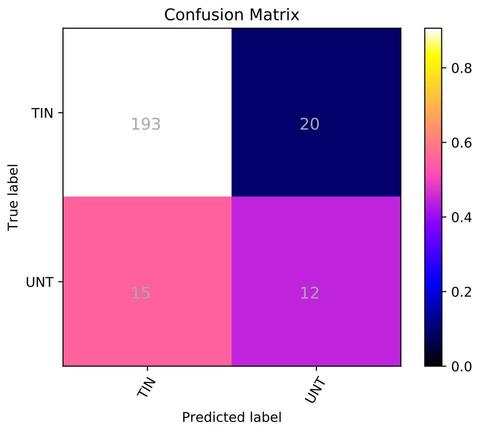


Figure 3: Confusion matrix of our best performed model for Sub-task B (biLSTM with specific configuration - Run3B)

The confusion matrix of our best performed model for sub-task C can be seen in Figure 4. It includes three classes as GRP (group), IND (individual) and OTH (other) for a sub-set of tweets containing targeted offense. The system achieves the highest precision for IND. The color range also shows the level of precision from darker to lighter.

Overall, we have achieved competitive results and rankings for each sub-task. Out of all our submissions, best performed ones for each sub-task and their comparison with the winner system of the shared task are given in Table 4.

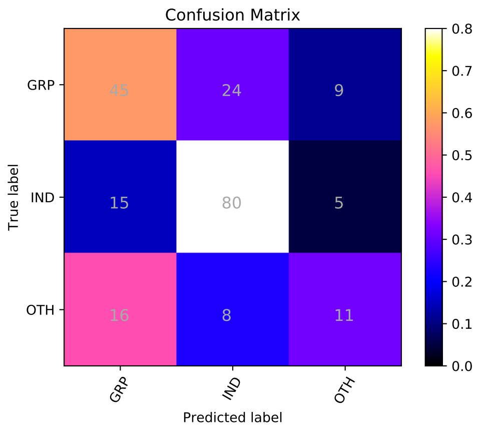


Figure 4: Confusion matrix of our best performed model for Sub-task C (biLSTM with specific configuration - Run3C)

5 Conclusion

In this paper, participation of LaSTUS/TALN to OffensEval: Identifying and Categorizing Offensive Language in Social Media (SemEval 2019 - Task 6) has been presented. We described and evaluated our system which is a bidirectional LSTM (biLSTM) model with an Attention layer on top, to classify if a tweet contains offensive language and the type and target of the offense for the offensive content.

For the future work, more detailed analyses on integration of linguistic annotations into neural network can be considered. In addition, a larger amount of data and also meta-data such as whether a tweet is a response to another tweet can represent contextual information and used to improve

System	F1 (macro)	Accuracy
All GRP baseline	0.1787	0.3662
All IND baseline	0.2130	0.4695
All OTH baseline	0.0941	0.1643
LaSTUS/TALN - Run1C	0.5631	0.6432
LaSTUS/TALN - Run2C	**0.5686**	**0.6385**
LaSTUS/TALN - Run3C	0.5480	0.6150

Table 3: Results of different submissions for **Sub-task C**.

	sub-task A	sub-task B	sub-task C
Best performer - F1(macro)	0.829	0.755	0.660
LaSTUS/TALN - F1(macro)	0.768	0.662	0.569
LaSTUS/TALN Ranking/Submissions	30 / 104	21 / 79	16 / 66

Table 4: Overall results and best rankings

performance.

Acknowledgements

Our work is partly supported by the Spanish Ministry of Economy and Competitiveness under the Maria de Maeztu Units of Excellence Programme (MDM-2015-0502). We thanks two reviewers for their constructive comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Francesco Barbieri, German Kruszewski, Francesco Ronzano, and Horacio Saggion. 2016. How cosmopolitan are emojis?: Exploring emojis usage and meaning over different languages with distributional semantics. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 531–535. ACM.

Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso. 2018. Evalita 2018: Overview on the 6th evaluation campaign of natural language processing and speech tools for italian. In *EVALITA@CLiC-it*.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in neural information processing systems*, pages 577–585.

T. Y. Chu, Kylie Jue, and Max L. Wang. 2017. Comment abuse classification with deep learning.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Michele P. Hamm, Amanda S. Newton, Annabritt Chisholm, Jocelyn Shulhan, Andrea Milne, Purnima Sundar, Heather Ennis, Shannon D. Scott, and Lisa Hartling. 2015. Prevalence and effect of cyberbullying on children and young people: A scoping review of social media studies. *JAMA pediatrics*, 169 8:770–7.

Cynthia Van Hee, Gilles Jacobs, Chris Emmery, Bart Desmet, Els Lefever, Ben Verhoeven, Guy De Pauw, Walter Daelemans, and Véronique Hoste. 2018. Automatic detection of cyberbullying in social media text. In *PloS one*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the germeval 2018 shared task on the identification of offensive language.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212.

LTL-UDE at SemEval-2019 Task 6: BERT and Two-Vote Classification for Categorizing Offensiveness

Piush Aggarwal, Tobias Horsmann, Michael Wojatzki and Torsten Zesch
Language Technology Lab
University of Duisburg-Essen
`piush.aggarwal@stud.uni-due.de,`
`{tobias.horsmann, michael.wojatzki, torsten.zesch}@uni-due.de`

Abstract

This paper describes LTL-UDE's systems for the *SemEval 2019 Shared Task 6*. We present results for Subtask A and C. In Subtask A, we experiment with an embedding representation of postings and use a Multi-Layer Perceptron and BERT to categorize postings. Our best result reaches the 10th place (out of 103) using BERT. In Subtask C, we applied a two-vote classification approach with minority fallback, which is placed on the 19th rank (out of 65).

1 Introduction

The Internet is frequently used for online debates and discussions, where individuals or groups are increasingly often verbally attacked. Online platform providers aim to remove such attacking posts or ideally, prevent them from being published. Manual verification of each posting by a human moderator is infeasible due to the high amount of postings created every day. Consequently, automated detection of such attacking postings is the only feasible way to counter this kind of hostility.

In this work, we present our results for the *SemEval 2019 Shared Task 6: Identifying and Categorizing Offensive Language in Social Media* (Zampieri et al., 2019b) on the OLID dataset (Zampieri et al., 2019a). Subtask A focuses on the binary distinction if a post is offensive or not, while Subtask C determines if the target is an individual, group, or other entity. Our submission for Subtask A ranks 10th, for Subtask C ranks 19th.

For Subtask A, we experiment with word list-based classification, using classifiers such as SVM or logistic regression based on sentence embeddings, and neural network-based models such as a Multi-layer Perceptron (MLP) and Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018). We find that the SVM performs best on our development set, but BERT reaches the best result on the test dataset. Moreover, a learning curve experiment suggests that more training data will lead only to minor improvements. In Subtask C, we choose a two-vote classification approach, where we let two systems compete with a fallback to the minority class in case the systems disagree. This fallback approach has a high robustness between our development and the official test dataset.

2 Related Work

Detection of offensive or potentially hurtful online postings is investigated under a variety of names. Waseem et al. (2017) focuses on *abusive language*, Kumar et al. (2018) tackles the problem as *aggression* while Macbeth et al. (2013) approaches this problem as *cyberbullying* to mention just a few. Furthermore, the field of hate speech detection is strongly related, which aims at detecting a similar kind of online statements (Waseem and Hovy, 2016; Wojatzki et al., 2018).

Common approaches to detecting such socially unacceptable statements utilize rich feature sets consisting of word ngrams, surface forms and syntactical features (Warner and Hirschberg, 2012; Nobata et al., 2016). Human-knowledge is provided by word lists containing offenses as key words or phrases (Bassignana et al., 2018; Wiegand et al., 2018b). Xiang et al. (2012) approaches the task as topic modelling problem using Latent Dirichlet Allocation (Blei et al., 2003).

These tasks are tackled with feature engineering-based approaches such as SVM or regression models but also with convolutional neural networks (Wiegand et al., 2018b).

3 Subtask A: Offensiveness

Subtask A is a binary classification task. A posting is either offensive or not offensive. For this task, we experiment with the following approaches:

Preprocessing We lowercase all postings and use the Ark Tokenizer (Gimpel et al., 2011) for word splitting. These preprocessing steps are used in all experiments.

Lexical Matching We use the following hand-crafted word lists of abusive words: (i) Profane Word List[1] containing more than 1,300 English tokens, (ii) UdS Lexicon of Abusive Words[2] having 1,651 entries (Wiegand et al., 2018a), and (iii) *Multilingual Lexicon of Words to Hurt* from HurtLex (Bassignana et al., 2018) with 9,313 terms.[3] A posting is classified as offensive if it contains any words in the before mentioned lists.

Posting Embeddings We represent each posting by a dense embedding, which we create from word embeddings by summing up the vector values of the word representations. The resulting posting vector is re-scaled into the range zero to one. We use the pre-trained embeddings provided by Mikolov et al. (2018), which are trained on the common crawl corpus.

Classifiers We apply the following classifiers: SVM (Chang and Lin, 2011), Logistic Regression (Fan et al., 2008), Random Forest (Breiman, 2001) and a Decision Tree (Breiman et al., 1984). We use the implementation provided by scikit-learn (Pedregosa et al., 2011) using default parameters.

Multi-Layer-Perceptron (MLP) With the same pre-processing and feature extraction steps used as for shallow models described above, we train a MLP with 100 hidden units in Scikit-Learn with ReLu as activation function and Adam optimizer (Kingma and Ba, 2014). We initialize the neural network with the *fasttext* word embeddings provided by Mikolov et al. (2018).

BERT We use the provided pre-trained BERT-base model (Devlin et al., 2018) to create a vector representation of a posting. We fine-tune the model on the training data set using a sequence-length of 128 and batches of 32. We also investigate the impact of enriching the training dataset with additional data by using machine translation. We back and forth translate the training data to obtain paraphrases of the original training data, which we expect to improve model performance. We translated the data into Russian, Chinese, and Arabic and back to English using Google's translation service. We repeated the fine-tuning with this enriched dataset.

Ensemble We combine the best three approaches (BERT, SVM, and Logistic Regression) in an ensemble, which was reported to often account for improvements in a similar shared task for German (Wiegand et al., 2018c). We use the majority vote of these classifiers as the prediction.

Set	Approach	F_1	Acc
dev	SVM	**.795**	**.814**
	BERT	.771	.799
	Ensemble	.767	.789
	BERT-trans	.732	.768
	Logistic Reg.	.704	.728
	MLP	.687	.705
	Random Forest	.641	.678
	Lexical Matching	.619	.680
	Decision Tree	.567	.585
	Baseline - all NOT	.400	.667
test	Ensemble	.748	.782
	BERT	**.798**	**.839**
	SVM	.729	.761
	Baseline - all NOT	.418	.720

Table 1: Subtask A: Results in term of macro F_1 on a held-back development dataset containing 1,048 offensive postings and 2,192 not offensive (NOT) ones.

3.1 Results

Table 1 shows the results for Subtask A. We report results on a self-created development dataset (25% of the original training data, 3,240 postings of which 1,048 postings are labeled as offensive and 2,192 as not offensive). We use the majority class as a baseline. On our dev dataset, we find that a SVM with the posting vector-representation achieves the best F-Score, followed by BERT. Contrary to our expectation, BERT's performance decreased by adding the machine-translated data. On the test dataset, we find BERT to perform best followed by the ensemble, which seems to add some additional robustness to the classification.

Learning curve A central question for shared tasks such as this one is if the amount of provided training data is sufficient to train a reliable clas-

[1] https://www.cs.cmu.edu/~biglou/resources/
[2] https://github.com/uds-lsv/lexicon-of-abusive-words
[3] http://hatespeech.di.unito.it/resources.html

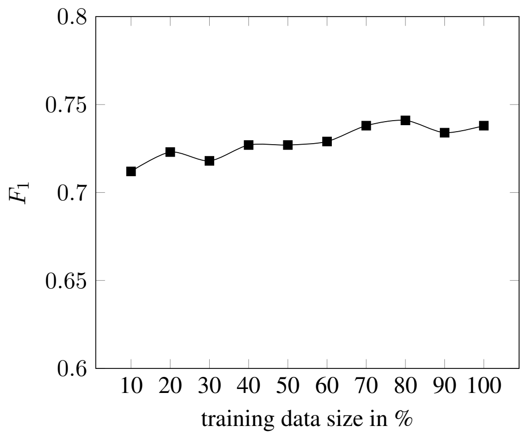


Figure 1: Learning curve on the training dataset. F-Score performance for adding an increasing amount of training data evaluated against the development set.

Set	Approach	F_1	Acc
dev	MLP+MLP	**.565**	.708
	SVM+MLP	.550	.688
	MLP+SVM	.541	.699
	SVM+SVM	.535	.701
	MLP	.523	.699
	SVM	.480	**.733**
	BERT	.492	.730
	Random Forest	..461	.676
	Decision Tree	.462	.604
	Logistic Regression	.508	.707
	Baseline - all IND	.255	.621
test	MLP+MLP	**.556**	**.666**
	SVM+MLP	.498	.615
	Baseline - all IND	.213	.469

Table 2: Subtask C results

sifier. Figure 1 shows a learning curve computed over the provided training data with testing against the hold-out development set. We split the training data into equal-sized data blocks which are randomly distributed over labels and add an increasing number of data blocks to see the performance improvement by adding more data. The results shows that improving the machine learning model is a more promising strategy than providing even more data as the slope indicates only minor improvements if more data is added.

4 Subtask C: Offense Targets

The goal in this subtask is to identify the kind of target at which a tweet is directed at (i.e. at this point it is already known that the tweet is a targeted offense, just the target itself is not yet determined). A target is either an individual (IND), a group (GRP), or other (OTH), if none of the previously mentioned two categories apply. We apply the same approaches as already used in Subtask A.

Two-Vote Classification with Minority Fallback Furthermore, an analysis of the class distribution showed that the class for *other* has comparatively few instances. This makes it challenging for a classifier to reliably detect such an under-represented class. Therefore, we attempt to redefine the problem as a binary classification problem using two classifiers. If the two classifiers agree in their prediction, we take the predicted class (either *individual* or *group*). In case of an disagreement, we select the minority class, *other*, as prediction. Thus, we also alter the training data to contain only two classes. The labels of the under-represented *other* class are mapped for one classifier to *individual* and for the other one to *group*, which creates a kind of minority-class noise. Our intuition is, if both classifier overcome the uncertainty added by the (small) amount of noise, the prediction is considered reliable. Consequently, we consider a disagreement as evidence for assigning the minority class.

Results Table 2 shows the results. We find that our two vote classification approach, using two MLPs, reaches the highest F-Score on the development and test set. On the development set, we reach the best accuracy result with an SVM but the considerably lower F-Score shows a strong bias towards a single class. Moreover, MLP+MLP shows a high robustness when comparing the F-Score performance between development and test set.

5 Conclusion

In this paper, we presented our approach on identifying and categorizing offensive language in social media. We mostly rely on lexical and semantic features for all subtasks. Results shows that semantic features have a significant impact on system performance. In general, our system leaves much room for improvement. Detection of offensiveness could probably benefit from more semantically oriented features that go beyond the surface form of words. We make the source code of our experiments publicly available[4].

[4] https://github.com/aggarwalpiush/OffensEval2019

References

Elisa Bassignana, Valerio Basile, and Viviana Patti. 2018. Hurtlex: A Multilingual Lexicon of Words to Hurt. In *Proceedings of the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018), Torino, Italy, December 10-12, 2018.*

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Leo Breiman. 2001. Random Forests. *Mach. Learn.*, 45(1):5–32.

Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. 1984. *Classification and regression trees*. The Wadsworth statistics/probability series. Wadsworth and Brooks/Cole Advanced Books and Software, Monterey, CA.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *J. Mach. Learn. Res.*, 9:1871–1874.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 42–47. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 1–11. Association for Computational Linguistics.

Jamie Macbeth, Hanna Adeyema, Henry Lieberman, and Christopher Fry. 2013. Script-based story matching for cyberbullying prevention. In *ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 901–906.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in Pre-Training Distributed Word Representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive Language Detection in Online User Content. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 145–153, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

William Warner and Julia Hirschberg. 2012. Detecting Hate Speech on the World Wide Web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Language Online*, pages 78–84. Association for Computational Linguistics.

Zeerak Waseem and Dirk Hovy. 2016. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93. Association for Computational Linguistics.

Michael Wiegand, Josef Ruppenhofer, Anna Schmidt, and Clayton Greenberg. 2018a. Inducing a Lexicon of Abusive Words – a Feature-Based Approach. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1046–1056. Association for Computational Linguistics.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018b. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval*.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018c. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language.

Michael Wojatzki, Tobias Horsmann, Darina Gold, and Torsten Zesch. 2018. Do Women Perceive Hate Differently: Examining the Relationship Between Hate Speech, Gender, and Agreement Judgments. In *Proceedings of the Conference on Natural Language Processing (KONVENS)*, pages 110–120, Vienna, Austria.

Guang Xiang, Bin Fan, Ling Wang, Jason Hong, and Carolyn Rose. 2012. Detecting Offensive Tweets via Topical Feature Discovery over a Large Scale Twitter Corpus. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1980–1984, New York, NY, USA. ACM.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

MIDAS at SemEval-2019 Task 6: Identifying Offensive Posts and Targeted Offense from Twitter

Haimin Zhang,[1] Debanjan Mahata,[1] Simra Shahid,[2] Laiba Mehnaz,[2]

Sarthak Anand,[3] Yaman Kumar,[5] Rajiv Ratn Shah,[4] Karan Uppal[1]

[1]Bloomberg, USA, [2]DTU-Delhi, India, [3]NSIT-Delhi, India, [4]IIIT-Delhi, India, [5]Adobe, India
hzhang449@bloomberg.net, dmahata@bloomberg.net, simrashahid_bt2k16@dtu.ac.in,
laibamehnaz@dtu.ac.in, sarthaka.ic@nsit.net.in, ykumar@adobe.com,
rajivratn@iiitd.ac.in, kuppal8@bloomberg.net

Abstract

In this paper, we present our approach and the system description for Sub-task A and Sub Task B of SemEval 2019 Task 6: Identifying and Categorizing Offensive Language in Social Media. Sub-task A involves identifying if a given tweet is offensive or not, and Sub Task B involves detecting if an offensive tweet is targeted towards someone (group or an individual). Our models for Sub-task A is based on an ensemble of Convolutional Neural Network, Bidirectional LSTM with attention, and Bidirectional LSTM + Bidirectional GRU, whereas for Sub-task B, we rely on a set of heuristics derived from the training data and manual observation. We provide a detailed analysis of the results obtained using the trained models. Our team ranked 5th out of 103 participants in Sub-task A, achieving a macro F1 score of 0.807, and ranked 8th out of 75 participants in Sub Task B achieving a macro F1 of 0.695.

1 Introduction

The unrestricted use of offensive language in social media is disgraceful for a progressive society as it promotes the spread of abuse, violence, hatred, and leads to other activities like trolling. Offensive text can be broadly classified as abusive and hate speech on the basis of the context and target of the offense. Hate speech is an act of offending, insulting or threatening a person or a group of similar people on the basis of religion, race, caste, sexual orientation, gender or belongingness to a specific stereotyped community (Schmidt and Wiegand, 2017; Fortuna and Nunes, 2018). Abusive speech categorically differs from hate speech because of its casual motive to hurt using general slurs composed of demeaning words. Both of them are the popular categories of offensive content, widespread in different social media channels.

With the democratization of the web, the usage of offensive language in online platforms is a clear indication of misuse of our right to 'Freedom of Speech'. While censorship of free moving online content curtails the freedom of speech, but unregulated opprobrious tweets discourage free discussions in the virtual world making the problem of identifying and filtering out offensive content from social media an important problem to be solved for creating a better society, both in and out of the Internet.

Detecting offensive content from social media is a hard research problem due to variations in the way people express themselves in a linguistically diverse social setting of the web. A major challenge in monitoring online content produced on social media websites like Twitter, Facebook and Reddit is the humongous volume of data being generated at a fast pace from varying demographic, cultural, linguistic and religious communities. Apart from the problem of information overload, social media websites pose challenges for automated information mining tools and techniques due to their brevity, noisiness, idiosyncratic language, unusual structure and ambiguous representation of discourse. Information extraction tasks using state-of-the-art natural language processing techniques, often give poor results when applied in such settings (Ritter et al., 2011). Abundance of link farms, unwanted promotional posts, and nepotistic relationships between content creates additional challenges. Due to the lack of explicit links between content shared in these platforms it is also difficult to implement and get useful results from ranking algorithms popularly used for web pages (Mahata et al., 2015).

Interests from both academia and industry has led to the organization of related workshops such as TA-COS[1], Abusive Language Online[2], and TRAC[3], along with shared tasks such as GermEval (Wiegand et al., 2018) and TRAC (Kumar et al., 2018). The task 6 of SemEval 2019 (Zampieri et al., 2019b) is one such recent effort containing short posts from tweets collected from the Twitter platform and annotated by human annotators with the objective of *identifying expressions of offensive language*, *categorization of offensive language* and *identifying the target against whom the offensive language is being used*, leading to three sub tasks (A, B and C). We only participate in two of them for which we define the problems.

Problem Definition Sub-task A - *Given a labeled dataset D of tweets, the objective of the task is to learn a classification/prediction function that can predict a label l for a given tweet t, where $l \in \{OFF, NOT\}$, OFF - denoting a tweet being offensive, and NOT - denoting a tweet being not offensive.*

Problem Definition Sub Task B - *Given a labeled dataset D of tweets, the objective of the task is to learn a classification/prediction function that can predict a label l for a given tweet t, where $l \in \{TIN, UNT\}$, TIN - denoting an offensive tweet targeted towards a group or an individual, and UNT - denoting a tweet that does not contain a targeted offense although it might use offensive language.*

Towards this objective we make the following contributions in this work:

- Train deep learning models of different architectures - Convolutional Neural Networks, Bidirectional LSTM with attention and Bidirectional LSTM + Bidirectional GRU, and report their results on the provided dataset. Our best model which ranked 5th in Sub-task A, is an ensemble of all the three deep learning architectures.

- We perform an analysis of the dataset, point out certain discrepancies in annotation and show how undersampling directed by error analysis could be sometimes useful for increasing the performance of the trained models.

[1] `http://ta-cos.org/`

[2] `https://sites.google.com/site/abusivelanguageworkshop2017/`

[3] `https://sites.google.com/view/trac1/home`

Next, we present previous works related to the task.

2 Related Work

Most of the previous works in this domain deals with the identification and analysis of the use of hate speech (Davidson et al., 2017), and abusive languages in online platforms (Nobata et al., 2016). Abusive speech categorically differs from hate speech because of its casual motive to hurt using general slurs composed of demeaning words. A proposal of typology of abusive language subtasks is presented in (Waseem et al., 2017). Both abusive as well as hate speech are sub-categories of offensive language. Detailed surveys of the works related to hate speech could be found in (Schmidt and Wiegand, 2017) and (Fortuna and Nunes, 2018).

One of the earliest efforts in hate speech detection can be attributed to (Spertus, 1997) who had presented a decision tree based text classifier for web pages with a 88.2 % accuracy. Contemporary works on Yahoo news pages were done (Sood et al., 2012), and later taken up by (Yin et al., 2016). (Xiang et al., 2012) detected offensive tweets using logistic regression over a tweet dataset with the help of a dictionary of 339 offensive words. Offensive text classification in online textual content have been tried previously for languages other than English, like German (Ross et al., 2017), Chinese (Su et al., 2017), Slovene (Fišer et al., 2017), Arabic (Mubarak et al., 2017), and in challenging cases of code-switched languages such as Hinglish (Mathur et al., 2018). However, despite the various endeavors by language experts and online moderators, users continue to disguise their abuse through creative modifications that contribute to multidimensional linguistic variations (Clarke and Grieve, 2017).

(Badjatiya et al., 2017) used CNN based classifiers to classify hateful tweets as racist and sexist. (Park and Fung, 2017) introduced a combination of CharCNN and WordCNN architectures for abusive text classification. (Gambäck and Sikdar, 2017) explored four CNN models trained on character n-grams, word vectors based on semantic information built using word2vec, randomly generated word vectors, and word vectors combined with character n-grams to develop a hate-speech text classification system. (Pitsilis et al., 2018) used an ensemble of RNNs in order to identify hateful content in social media.

Some of the recent works in this domain has been on identifying profanity vs. hate speech (Malmasi and Zampieri, 2018), which highlights the challenges of distinguishing between profanity, and threatening language which may not actually contain profane language. On a similar direction there has been work on understanding the main intentions behind vulgar expressions in social media (Holgate et al., 2018). Various approaches have been taken to tackle both textual as well as multimodal data from Twitter and social media in general, in order to build deep learning classifiers for similar tasks (Baghel et al., 2018; Kapoor et al., 2018; Mahata et al., 2018a,b; Jangid et al., 2018; Meghawat et al., 2018; Shah and Zimmermann, 2017).

3 Dataset

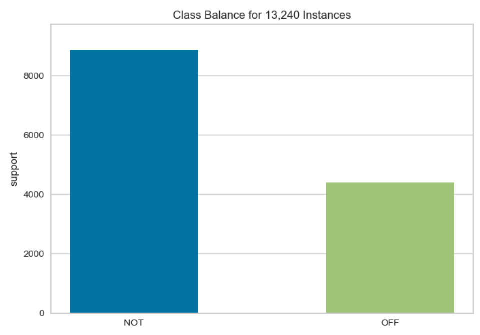


Figure 1: Distribution of classes (OFF - Offensive and NOT - Not Offensive) for Sub-task A.)

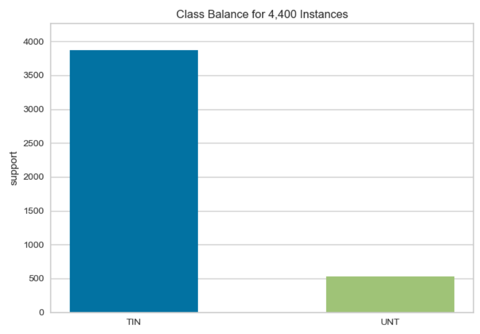


Figure 2: Distribution of classes (TIN - Targeted Offense and UNT - Untargeted Offense) for Sub Task B.)

The dataset provided for the tasks was collected through Twitter API by searching for tweets containing certain selected keyword patterns popular in offensive posts. Around 50% of the keyword patterns were political in nature such as 'MAGA', 'antifa', 'conservative' and 'liberal'. The other half were based on keyword patterns such as 'he is', 'she is', in combination with metadata provided by the Twitter API that marks a tweet to be 'unsafe'. The annotation of the collected data was done using figure eight, which is a popular crowdsourcing platform. 14,100 tweets were selected in the final dataset with 13,240 provided as the training data and 860 as the test data. The details of the dataset, its collection process and annotation agreements could be found in (Zampieri et al., 2019a).

Figures 1 and 2, shows the distribution of the classes in the subsets of the data provided for Subtask A and Sub Task B, respectively. The distributions show the imbalance in class labels. We also took a detailed look at the dataset and found discrepancies between the definition of the classes as provided by the organizers and the actual annotations. The mislabeling was more prominent as an offensive post being labeled as not offensive. We observed such wrong annotations when performing manual error analysis on the predictions provided by an initially trained classifier, which was a simple Convolutional Neural Network. About 4 % of the posts seemed to have been mislabeled, which we found through manual inspection and removed them from the training data. Here are few such examples.

- @user @user @user @user @user @user @user what a stupid incompetent devious and toxic pm ! may haven't you forgotten 17.4 million voters ? betray us at your peril ! you are eroding faith in democracy + destroying tory party ! you should go url. (**Original Label: NOT**)

- angelina is so funny at rhe wrong times imngonna shoot this bitch uppdoals. (**Original Label: NOT**)

- @user @user so and accusation by a libtarded trump hating liberal activist against a trump appointee doesnt make u wonder if the accusation was politically motivated in the slightest ? no ? this is why conservatives think u are all stupid . because u are . (**Original Label: NOT**)

This increased the performances of our trained models and could be considered as a heuristic based undersampling of the provided dataset.

4 Experiments and Results

We train different deep learning models for the Sub-task A and rely on heuristics learnt from the training data for Sub-task B. In this section we explain the steps taken for pre-processing data and training the predictive models and give a short description of the heuristics that we came up with after analyzing the data.

4.1 Data Preprocessing

Before feeding the dataset to any machine learning model we took some steps to process the data. For all our experiments we used Keras[4] as the machine learning coding library. Some of the preprocessing steps that we took are:

Tokenization - Tokenization is a fundamental preprocessing step and could be one of the important factors influencing the performance of a machine learning model that deals with text. As tweets include wide variation in vocabulary and expressions such as user mentions and hashtags, the tokenization process could become a challenging task. We used the nltk's[5] tweet tokenizer in order to tokenize the tweets provided in the dataset by overriding the default tokenizer provided in keras.

Cleaning and Normalization - Normalization of tokens were also done using some hand-crafted rules. The # symbol was removed from the tweets along with mapping few popular offensive words to a standard form. For example, 'bi*ch', 'b**ch', 'bi**h', 'biatch' were all mapped to 'bitch', and 'sob', 'sobi*ch', were mapped to 'son of bitch'. The @user tokens were removed. The hashtags that contained two or more words were segmented into their component words. For example #fatbastard was converted to fat bastard.

4.2 Training Deep Learning Models

In order to train deep learning models we need to provide the input as a matrix and the input words need to be mapped to their embeddings which provides richer semantic representation of words in comparison to the one-hot vectors. Each tweet is treated as a sequence of words and may vary in their lengths. We fix 200 as the max length and pad the input sequences in order to make their lengths fixed to 200. For, our experiments we used the 200 dimensional Glove embeddings[6] trained on tweets and 400 dimensional Godin embeddings[7]. There was no significant difference in the results while training our initial models by using one over the other. Therefore for all our models as presented in this work we selected the Glove embeddings as the pre-trained word embedding of our choice due to its lower dimensions resulting in lesser training of weights in the neural network.

We train the following architectures for Sub-task A having the parameters as explained next.

Convolutional Neural Network - Convolutional neural networks are effective in text classification tasks primarily because they are able to pick out salient features (e.g., tokens or sequences of tokens) in a way that is invariant to their position within the input sequence of words. In our model, we use three different filters with sizes 2, 3 and 4. For each filter size, 256 filters are used. A max pooling layer is then applied for each filter size. The resultant vectors are concatenated to form the vector that represents the whole tweet. A drop out layer with drop out rate 0.3 is applied before the input to the Multi Layer Perceptron with 256 neurons for classification. We also use a dropout layer after the embedding with dropout rate 0.3 to randomly drop words, which we find helpful to resolve overfitting issue. Sigmoid activation function is applied to the final layer.

Bidirectional LSTM with Attention - Bidirectional LSTM (BLSTM) is an extension of LSTM in which two LSTM models are trained on the input sequence. The first on the input sequence as-is and the second on its reversed copy. This can provide additional context to the network and result in faster and sometimes better learning. They have shown very good results in sequence classification tasks. We use 64 LSTM units with 0.2 drop out, one attention layer is added on the sequence of result vectors from BLSTM. 128 neurons are used in the final Multi Layer Perceptron layer for classification. Sigmoid activation function is applied to the final layer.

Bidirectional LSTM followed by Bidirectional GRU - We use 64 LSTM units wrapped by a Bidirectional layer, 0.3 was the dropout rate, followed by a Bidirectional GRU with 64 GRU units also with 0.3 dropout. Then a max pooling and average pooling are used and concatenated before input to the final Multi Layer Perceptron layer with 128 neurons for classification. Sigmoid activation

[4]https://keras.io/

[5]https://www.nltk.org/api/nltk.tokenize.html

[6]https://github.com/plasticityai/magnitude

[7]https://fredericgodin.com/software/

System	F1 (macro)	Accuracy
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
Convolutional Neural Network (on training data)	0.8020	0.8387
Bidirectional LSTM with Attention (on training data)	0.7851	0.8246
Bidirectional LSTM + Bidirectional GRU (on training data)	0.7893	0.8301
MIDAS Submission 1 on test data (CNN)	0.7964	0.8395
MIDAS Submission 2 on test data (Ensemble of CNN, BLSTM with Attention, BLSTM + BGRU)	0.8066	0.8407

Table 1: Results for Sub-task A.

System	F1 (macro)	Accuracy
All TIN baseline	0.4702	0.8875
All UNT baseline	0.1011	0.1125
MIDAS Submission 1	0.6952	0.8667

Table 2: Results for Sub-task B.

function is applied to the final layer.

For all three models we add a drop out layer after the embedding to randomly drop words, which we find helpful to address overfitting issue, and early stop is used with restoring the best model weights. Grid search is used to find the best parameters for each model. Table 1 presents the performance of each of these networks on the modified dataset as already explained in Section 3.

Often, one solution to a complex problem does not fit to all scenarios. Thus, researchers use ensemble techniques to address such problems. Historically, ensemble learning has proved to be very effective in most of the machine learning tasks including the famous winning solution of the Netflix Prize. Ensemble models can offer diversity over model architectures, training data splits or random initialization of the same model or model architectures. Multiple average or low performing learners are combined to produce a robust and high performing learning model. We do the same in our experiments. We combine the trained deep learning models having different architectures as an ensemble by averaging their final predictions. We had also tried the stacked ensemble approach as explained in (Mahata et al., 2018b). But it didn't give promising results in first few iterations. Moreover, it was computationally expensive and due to lack of sufficient time we, did not go further in that route.

Our ensemble model performed better than the individual models and was also submitted to the competition, which was finally ranked 5th amongst 103 participants. Figure 3 presents the confusion matrix of our submission for Sub-task A. Some of the samples from the training dataset,

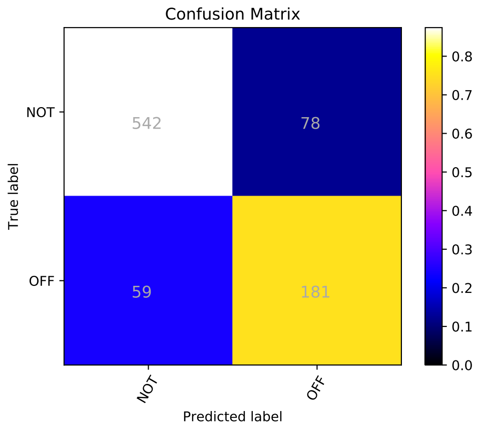


Figure 3: Confusion Matrix for MIDAS submission 2 for Sub-task A.

which were very hard for our final model to predict are:

- More like #Putin every day. #MAGA URL (OFF)
- @USER Hitler would be so proud of David Hogg trying to disarm American citizen so when Democrats come to power-we are helpless And cannot defend ourselves-; that's why we have they AR15's (NOT)
- @USER good job (sarcasm). Also great they have gun control laws its saving lives! (More sarcasm). (OFF)

4.3 Heuristics for Sub-task B

Due to lack of time from our part, we were not able to train good machine learning models for Sub-task B. The preliminary models that we trained showed performances that was similar to that of a random model biased by the class distribution of the training data. The training dataset for Sub-task B was highly imbalanced which was a major challenge. We would like to have an in depth look at Sub-task B in the near future.

For the sake of submission to the competition we came up with certain heuristics in order to decide whether an offensive post is targeted or not.

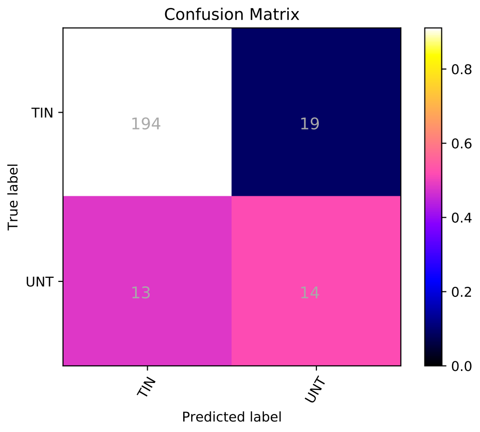


Figure 4: Confusion Matrix for MIDAS submission 1 for Sub-task B.

We skipped the pre-processing part of the tweets that we did before training the machine learning models as described in Section 4.1. We looked at the frequency distribution of words and hashtags in the training dataset as well as observed the patterns of the posts. After doing that we did find that some of the hashtags like '#maga', '#liberals', '#kavanaugh', '#qanon', etc were frequently occurring. and so are some of the tokens like 'antifa', 'president', 'trump', 'potus', 'liberals', 'conservatives', 'democrat', 'nigga', 'gay', 'jew'. Top 100 such tokens and hashtags were compiled after eliminating some of them manually if they didn't make any sense, for example some unwanted stop words. We also extracted POS tags of the tweets using TweeboParser[8] and extracted named entities (only PERSON, ORG, LOCATION, FACILITY) using SpaCy[9]. We framed our final heuristic based on the following rules:

- If the post includes any of the 100 hashtags then it is considered as targeted offense (TIN).
- else if the post includes any of the 100 tokens then it is considered as targeted offense (TIN).
- else if no named entity in the post and no Personal Pronoun and Proper Nouns are present in the post then it is a untargeted offense (UNT).
- else if the post has he/she is, you are, he she then it is considered as targeted offense (TIN).
- else if the post has pattern ' Starts with hashtag followed by verbs and named entity' then it is considered as targeted offense (TIN).
- else If there is a named entity then it is considered as targeted offense (TIN).
- all other cases are considered as untargeted offense (UNT).

We do not think this to be a robust model and it was only possible to come up with the heuristics because there were certain patterns in the dataset that was very obvious to bare human eye. Given that the dataset is very small, these heuristics can never scale well. One of the reasons behind discovering such patterns could also be because of the way the dataset was collected. Now that we know how it was collected as explained in Section 3, these patterns make more sense and it does explain why we could perform reasonably well even though we came up with such naive patterns in haste. Figure 4 presents the confusion matrix of our submission for Sub-task B and Table 2 presents the performance on the test dataset.

5 Conclusion and Future Work

In this work, we report our models and their respective performances in Sub-task A and B of SemEval-2019 Task 6 OffensEval: Identifying and Categorizing Offensive Language in Social Media. We showed how an ensemble of deep learning models performed well in the provided dataset and was ranked 5th in the competition in Sub-task A. Due to the inherent biases in collecting the dataset we believe that we were able to come up with naive heuristics for Sub-task B and was able to rank 8th in the competition.

In the future we would like to solve Sub-task B using a machine learning approach. We would also like to look at other machine learning architectures and ensemble methods for the different sub tasks in the competition. Out of three sub tasks, we were able to attempt only two of them. In the near future we would like to tackle the problem posed in Sub-task C. Some of the other areas that could be explored are cleaning the dataset by correcting the annotations and studying the problem of inherent biases that can occur in samples collected based on keyword patterns.

[8]http://www.cs.cmu.edu/ ark/TweetNLP/
[9]https://spacy.io

References

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.

Nupur Baghel, Yaman Kumar, Paavini Nanda, Rajiv Ratn Shah, Debanjan Mahata, and Roger Zimmermann. 2018. Kiki kills: Identifying dangerous challenge videos from social media. *arXiv preprint arXiv:1812.00399*.

Isobelle Clarke and Jack Grieve. 2017. Dimensions of abusive language on twitter. In *Proceedings of the first workshop on abusive language online*, pages 1–10.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Darja Fišer, Tomaž Erjavec, and Nikola Ljubešić. 2017. Legal Framework, Dataset and Annotation Schema for Socially Unacceptable On-line Discourse Practices in Slovene. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Eric Holgate, Isabel Cachola, Daniel Preoţiuc-Pietro, and Junyi Jessy Li. 2018. Why swear? analyzing and inferring the intentions of vulgar expressions. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4405–4414.

Hitkul Jangid, Shivangi Singhal, Rajiv Ratn Shah, and Roger Zimmermann. 2018. Aspect-based financial sentiment analysis using deep learning. In *Companion of the The Web Conference 2018 on The Web Conference 2018*, pages 1961–1966. International World Wide Web Conferences Steering Committee.

Raghav Kapoor, Yaman Kumar, Kshitij Rajput, Rajiv Ratn Shah, Ponnurangam Kumaraguru, and Roger Zimmermann. 2018. Mind your language: Abuse and offense detection for code-switched languages. *arXiv preprint arXiv:1809.08652*.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Debanjan Mahata, Jasper Friedrichs, Rajiv Ratn Shah, and Jing Jiang. 2018a. Detecting personal intake of medicine from twitter. *IEEE Intelligent Systems*, 33(4):87–95.

Debanjan Mahata, Jasper Friedrichs, Rajiv Ratn Shah, et al. 2018b. # phramacovigilance-exploring deep learning techniques for identifying mentions of medication intake from twitter. *arXiv preprint arXiv:1805.06375*.

Debanjan Mahata, John R Talburt, and Vivek Kumar Singh. 2015. From chirps to whistles: discovering event-specific informative content from twitter. In *Proceedings of the ACM web science conference*, page 17. ACM.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Puneet Mathur, Rajiv Shah, Ramit Sawhney, and Debanjan Mahata. 2018. Detecting offensive tweets in hindi-english code-switched language. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, pages 18–26.

Mayank Meghawat, Satyendra Yadav, Debanjan Mahata, Yifang Yin, Rajiv Ratn Shah, and Roger Zimmermann. 2018. A multimodal approach to predict social media popularity. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 190–195. IEEE.

Hamdy Mubarak, Kareem Darwish, and Walid Magdy. 2017. Abusive language detection on arabic social media. In *Proceedings of the First Workshop on Abusive Language Online*, pages 52–56.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive Language Detection in Online User Content. In *Proceedings of the 25th International Conference on World Wide Web*, pages 145–153. International World Wide Web Conferences Steering Committee.

Ji Ho Park and Pascale Fung. 2017. One-step and two-step classification for abusive language detection on twitter. *arXiv preprint arXiv:1706.01206*.

Georgios K Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1524–1534. Association for Computational Linguistics.

Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Wojatzki. 2017. Measuring the reliability of hate speech annotations: The case of the european refugee crisis. *arXiv preprint arXiv:1701.08118*.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Rajiv Shah and Roger Zimmermann. 2017. *Multimodal analysis of user-generated multimedia content*. Springer.

Sara Owsley Sood, Elizabeth F Churchill, and Judd Antin. 2012. Automatic identification of personal insults on social news sites. *Journal of the American Society for Information Science and Technology*, 63(2):270–285.

Ellen Spertus. 1997. Smokey: Automatic recognition of hostile messages. In *AAAI/IAAI*, pages 1058–1065.

Huei-Po Su, Chen-Jie Huang, Hao-Tsung Chang, and Chuan-Jie Lin. 2017. Rephrasing Profanity in Chinese Text. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval*.

Guang Xiang, Bin Fan, Ling Wang, Jason Hong, and Carolyn Rose. 2012. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1980–1984. ACM.

Dawei Yin, Yuening Hu, Jiliang Tang, Tim Daly, Mianwei Zhou, Hua Ouyang, Jianhui Chen, Changsung Kang, Hongbo Deng, Chikashi Nobata, et al. 2016. Ranking relevance in yahoo search. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 323–332. ACM.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Nikolov-Radivchev at SemEval-2019 Task 6: Offensive Tweet Classification with BERT and Ensembles

Victor Radivchev
Sofia University, FMI
victor.radivchev@gmail.com

Alex Nikolov
Sofia University, FMI
alexnickolow@gmail.com

Abstract

This paper examines different approaches and models towards offensive tweet classification which were used as a part of the OffensEval 2019 competition. It reviews Tweet preprocessing, techniques for overcoming unbalanced class distribution in the provided test data, and comparison of multiple attempted machine learning models.

1 Introduction

The purpose of this paper is to explore different approaches towards classifying tweets based on whether they are offensive or not, whether offensive tweets are targeted, and identifying the target group of offensive tweets either an individual, a group, or other. Those are the terms of the OffensEval 2019 competition in which we participated. Each of the described activities constituted a separate subtask from the competition. A maximum of three submissions were allowed per subtask which required careful preliminary analysis of the model results during the training phase. A training set of over 13,000 tweets, containing labels for all three subtasks. Each of the subtasks was scored using macro F1 score.

2 Related Work

One of the most effective strategies for tackling this problem is to use computational methods to identify offense, aggression, and hate speech in user-generated content (e.g. posts, comments, microblogs, etc.). This topic has attracted significant attention recently as evidenced in publications from the last two years.

Survey papers describing key areas that have been explored for this task include (Schmidt and Wiegand, 2017), (Fortuna and Nunes, 2018) and (Malmasi and Zampieri, 2017). The dataset for this competition is explained in (Zampieri et al., 2019a) and different approaches to the same problem are reported in (Zampieri et al., 2019b).

In order to classify correctly abusive language it is important to analyze its types. A proposal of typology of abusive language sub-tasks is presented in (Waseem et al., 2017) and (ElSherief et al., 2018) examines the target of the speech: either directed towards a specific person or entity, or generalized towards a group of people sharing a common protected characteristic. (Fišer et al., 2017) proposes a legal framework, dataset and annotation schema of socially unacceptable discourse practices on social networking platforms in Slovenia. Finally, a recent discussion on identifying profanity vs. hate speech is presented in (Malmasi and Zampieri, 2018). This work highlighted the challenges of distinguishing between profanity, and threatening language which may not actually contain profane language.

Approaches to detecting hate speech on Twitter using convolutional neural networks and convolution-GRU based deep neural network are discussed in (Gambäck and Sikdar, 2017) and (Zhang et al., 2018) respectively.

Additional related work is presented in workshops such as TA-COS[1], Abusive Language Online[2], and TRAC[3] and related shared tasks such as GermEval (Wiegand et al., 2018) and TRAC (Kumar et al., 2018).

[1]http://ta-cos.org/
[2]https://sites.google.com/site/abusivelanguageworkshop2017/
[3]https://sites.google.com/view/trac1/home

3 Methodology and Data

The data was split into a training and validation set in a ratio of 10:1. All tasks had similar preprocessing and multiple models were trained on the training set. Depending on their performance on the validation set each time the best 3 were submitted.

3.1 Preprocessing

We started our tweet preprocessing by removing most punctuation marks which do not include any useful information for text classification. The symbols '@' and '#' were excluded from the list due to their specific semantics in tweets. Afterwards the tweets were subjected to tokenization and lowercasing.

All occurrences of tokens beginning with a hashtag were split into the separate words comprising the token, provided that each separate word is uppercased. For example, the token #HelloThere is split into two tokens hello and there.

Afterwards we proceeded with removing a variety of different stop words. When training models for the second and third subtask, we excluded personal and possessive pronouns from the list of stop words, as they can contain valuable information for classifying a tweet as targeted or not, or identifying the target group of a targeted tweet. We also attempted lemmatization and spell correction but the results were slightly worse or on par with the ones achieved without using these two techniques.

Pre-trained word vectors on Twitter from project GloVe (Pennington et al., 2014) were used for encoding words to a vector space. Four different vector dimensions were available for use 25, 50, 100, and 200. Although results were slightly better when using higher dimensional vectors, using 200-dimensional vectors proved to have no significant advantage in achieved results over 100-dimensional ones, and proved to be more computationally expensive, which lead us to use 100-dimensional vectors for each subtask.

3.2 Models

We trained a large variety of different models and combined the best of them in ensembles. For all models the embedding layer was freezed, becaused that proved less prone to overfitting.

- Standard Nave Bayes and Support Vector Machine (SVM) from scikit-learn library in python.
- Convolutional Neural Network (CNN) with GlobalMaxPooling and hidden dense layer on top.
- Multilayer Perceptron Network (MLP) with two hidden layers.
- FastText models with n-grams of size 2.
- Recurrent Neural Network (RNN) with GRU units and attention layer and hidden dense layer on top.
- Deep Pyramid Convolutional Neural Network (DPCNN) (Johnson and Zhang, 2017).
- Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018).
- Soft Voting Classifier (SVC) - averages the predictions of the single models.
- Logistic Regression - meta model trained on half of the validation set with predictions from the single classifiers as features.

3.3 Class imbalance

One of the challenges of the competition was the imbalance of classes for the second and third subtask. We experimented with different techniques for overcoming this challenge:

- Oversampling duplicating some of the examples from the poorly represented classes.
- Class weights assigning lower weights to examples from classes which are better represented and higher weights to examples from classes with a lower overall count.
- Modification of the thresholds used for classifying an example. For example, for a standard binary classification a threshold of 0.5 is applied to the predicted probability in order

to distinguish between the two classes. We attempted to lower this threshold to different levels.

For all model apart from BERT the class weight option was chosen. Only for BERT on subtask C the thresholds were changed instead. For classes OTH and GRP we used thresholds of 0.2 and 0.3 respectively and if any of them was exceeded we would directly assign that class. If both were exceeded we would assign OTH as the class. The coefficients were derived via cross-validation.

4 Results

The results from the test sets for each subtask are displayed below. We have also provided the results from our validation sets, those were the basis upon which we decided which models predictions to submit.

The individual model with the best performance on subtask A was BERT-Large, Uncased with a macro F1 score of 0.781 on the validation set and was selected as one of the models for submission. The other two submitted models were the soft voting classifier with score of 0.788 and the logistic regression model 0.800. The scores of the other trained models are displayed below.

System	F1 (macro)
Logistic Regression	**0.800**
SVC	0.788
BERT-Large	0.781
RNN	0.773
DPCNN	0.768
CNN	0.765
FastText	0.759
Nave Bayes	0.744
MLP	0.742
SVM	0.705

Table 1: Results on the validation set for Sub-task A.

The ensemble models proved to have overfit on the training data and out of the models we have submitted BERT had the highest score, ranking second overall amongst all participants.

In subtask B the highest scoring models on the validation set was the soft voting classifier with a score of 0.64, closely followed by RNN and CNN 0.63. BERT-Base, Uncased performed surprisingly poorly and achieved a score of 0.59.

The soft voting classifier scored the highest on the test set and ranked 16th overall.

System	F1 (macro)	Accuracy
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
SVC	0.7252	0.8105
Logistic Regression	0.7867	0.8453
BERT-Large	**0.8153**	**0.8547**

Table 2: Results on the test set for Sub-task A.

System	F1 (macro)
SVC	**0.642**
RNN	0.633
CNN	0.631
DPCNN	0.630
Logistic Regression	0.629
MLP	0.614
FastText	0.612
BERT-Base	0.599
Nave Bayes	0.596
SVM	0.576

Table 3: Results on the validation set for Sub-task B.

In subtask C BERT-Base, Uncased was by far the best individual model, achieving a score of 0.64, surpassing its closest contender (Multi-Layered Perceptron) by approximately 0.045. The third model which we submitted was the soft voting classifier with a score of 0.60.

BERT significantly outperformed every other submitted model, securing us first place in subtask C.

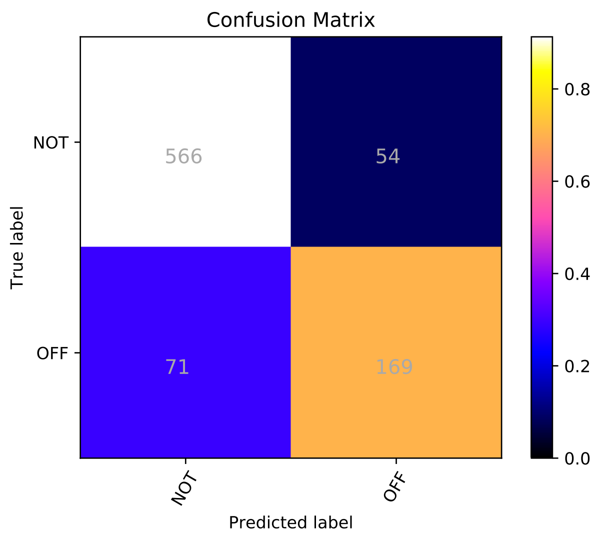


Figure 1: Sub-task A, vradivchev anikolov CodaLab BERT

System	F1 (macro)	Accuracy
All TIN baseline	0.4702	0.8875
All UNT baseline	0.1011	0.1125
RNN	0.6354	0.7667
SVC	**0.6674**	**0.8208**
CNN	0.6248	0.7833

Table 4: Results on the test set for Sub-task B.

System	F1 (macro)
BERT-Base	**0.644**
SVC	0.603
MLP	0.595
Logistic Regression	0.590
RNN	0.586
CNN	0.571
FastText	0.570
DPCNN	0.568
Nave Bayes	0.567
SVM	0.546

Table 5: Results on the validation set for Sub-task C.

System	F1 (macro)	Accuracy
All GRP baseline	0.1787	0.3662
All IND baseline	0.2130	0.4695
All OTH baseline	0.0941	0.1643
BERT-Base	**0.6597**	**0.7277**
MLP	0.5591	0.6808
SVC	0.6107	0.6948

Table 6: Results on the test set for Sub-task C.

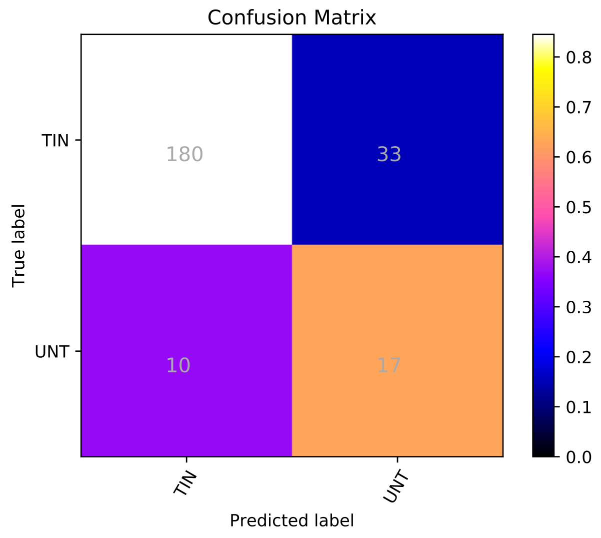


Figure 2: Sub-task B, vradivchev anikolov CodaLab Soft Voting Classifier

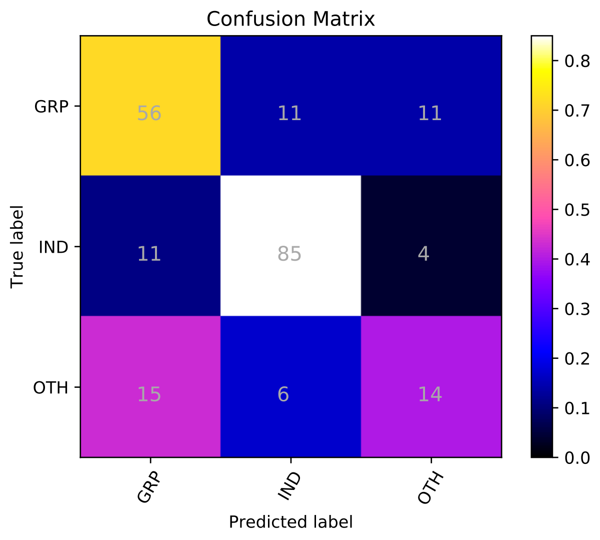


Figure 3: Sub-task C, vradivchev anikolov CodaLab BERT

5 Conclusion

Google's BERT model proved to be a powerful tool for text classification. Not only did it outperform common models on the validation set, but based on the results from the test set it did so without overfitting on the data.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Darja Fišer, Tomaž Erjavec, and Nikola Ljubešić. 2017. Legal Framework, Dataset and Annotation Schema for Socially Unacceptable On-line Discourse Practices in Slovene. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *ACL*.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 467–472.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

NIT_Agartala_NLP_Team at SemEval-2019 Task 6: An Ensemble Approach to Identifying and Categorizing Offensive Language in Twitter Social Media Corpora

Steve Durairaj Swamy[1], Anupam Jamatia[1], Björn Gambäck[2] and Amitava Das[3]

[1]National Institute of Technology, Agartala, India
[2]Norwegian University of Science and Technology, Trondheim, Norway
[3]Mahindra École Centrale,Hyderabad, Telangana, India

{steve050798,anupamjamatia}@gmail.com,gamback@ntnu.no,amitava.das@mechyd.ac.in

Abstract

The paper describes the systems submitted to OffensEval (SemEval 2019, Task 6) on 'Identifying and Categorizing Offensive Language in Social Media' by the 'NIT_Agartala_NLP_Team'. A Twitter annotated dataset of 13,240 English tweets was provided by the task organizers to train the individual models, with the best results obtained using an ensemble model composed of six different classifiers. The ensemble model produced macro-averaged F_1-scores of 0.7434, 0.7078 and 0.4853 on Subtasks A, B, and C, respectively. The paper highlights the overall low predictive nature of various linguistic features and surface level count features, as well as the limitations of a traditional machine learning approach when compared to a Deep Learning counterpart.

1 Introduction

Offensive language has been the scourge of the internet since the rise of social media. Social media provides a platform for everyone and anyone to voice their opinion. This has empowered people to make their voices heard and to speak out on global issues. The downside to this, however, is the misuse of such platforms to attack an individual or a minority group, and to spread hateful opinions. Pairing this with the perceived anonymity the internet provides, there has been a massive upswing in the use of social media for cyberbullying and hate speech, with technology giants coming under increased pressure to address the issue.

Most of what we may be interested in detecting can be broadly labelled as hate speech, cyberbullying or abusive use of swearing. The union of these three subsets form what can be identified as 'Offensive Language on Social Media'. However, what we consider offensive is often a grey area, as is evident by the low inter-annotator agreement rates when labelling data for offensive language (Waseem et al., 2017b).

Detecting offensive language has proven to be difficult, due to the broad spectrum in which language can be used to convey an insult. The nature of the abuse can be implicit — drawing from sarcasm and humour rather than offensive terms — as well as explicit, by making extensive use of traditional offensive terms and profanity. It does not help that the reverse is also entertained, with profanity often being used to imply informality in speech or for emphasis. Coincidentally, these are also the reasons why lexical detection methods have been unfruitful in classifying text as offensive or non-offensive.

The OffensEval 2019 shared task (Zampieri et al., 2019b) is one of several endeavours to further the state-of-the-art in addressing the offensive language problem. The paper describes the insights obtained when tackling the shared task using an ensemble of traditional machine learning classification models and a Long Short-Term Memory (LSTM) deep learning model. Section 2 first discusses other related approaches to detecting hate speech and offensive language. Then Section 3 describes the dataset and Section 4 the ideas and methodology behind our approach. Section 5 reports the results obtained, while Section 6 discusses those results with a particular eye towards the errors committed by the models. Finally, Section 7 sums up the key results and points to ways the work can be extended.

2 Related Work

Most datasets for offensive language detection represent multiclass classification problems (Davidson et al., 2017; Founta et al., 2018; Waseem and Hovy, 2016), with the annotations often obtained via crowd-sourcing portals, with varying degrees of success. Waseem et al. (2017b) state that annotation via crowd-sourcing tends to work best when the abuse is explicit (Waseem and Hovy, 2016), but is considerably less reliable when considering implicit abuse (Dadvar et al., 2013; Justo et al., 2014; Dinakar et al., 2011). They propose a typology that can synthesise different offensive language detection subtasks. Zampieri et al. (2019a) expand on these ideas and propose a hierarchical three-level-annotation model, which is used in the OffensEval 2019 shared task. Another issue is whether the datasets should be balanced or not (Waseem and Hovy, 2016), since there are much fewer offensive comments than benign comments in randomly sampled real-life data (Schmidt and Wiegland, 2017).

Classical Machine learning algorithms have been wielded to some success in automated offensive language detection, mainly Logistic Regression (Davidson et al., 2017; Waseem and Hovy, 2016; Burnap and Williams, 2015) and Support Vector Machines (Xu et al., 2012; Dadvar et al., 2013). Recently, however, deep learning models have outperformed their traditional machine learning counterparts, with both Recurrent Neural Networks (RNN) — such as LSTM (Pitsilis et al., 2018) and Bi-LSTM (Gao and Huang, 2017) — and Convolutional Neural Networks (CNN) having been used. Gambäck and Sikdar (2017) utilised a CNN model with word2vec embeddings to obtain higher F_1-score and precision than a previous logistic regression model (Waseem and Hovy, 2016), while Zhang et al. (2018) combined a CNN model with a Gated Recurrent Unit (GRU) layer. Malmasi and Zampieri (2018) used an ensemble system much like ours to separate profanity from hate speech, but reported no significant improvement over a single classifier system.

In terms of features, simple bag of words models have proven to be highly predictive (Waseem and Hovy, 2016; Davidson et al., 2017; Nobata et al., 2016; Burnap and Williams, 2015). Mehdad and Tetreault (2016) endorsed the use of character n-grams over token n-grams citing their ability to glaze over the spelling errors that are frequent in online texts. Nobata et al. (2016); Chen et al. (2012) showed small improvements by including features capturing the frequency of different entities such as URLs and mentions, with other features such as part-of-speech (POS) tags (Xu et al., 2012; Davidson et al., 2017) and sentiment scores (Van Hee et al., 2015; Davidson et al., 2017) also having been used (Schmidt and Wiegland, 2017). More recently, meta information about the users have been suggested as features, but no consistent correlation between user information and tendency for offensive behaviour online has been shown, with Waseem and Hovy (2016) claiming gender information leading to improvements in classifier performance, but with Unsvåg and Gambäck (2018) challenging this and reporting user-network data to be more important instead. Wulczyn et al. (2017) concluded that anonymity leads to an increase in the likelihood of a comment being an attack.

3 Data

The training dataset used for the shared task, the Offensive Language Identification Dataset (Zampieri et al., 2019a), contains 13,240 tweets, with each tweet having been annotated on the basis of a hierarchical three-level model. An additional 860 tweets were used as the test set for the shared task. The three levels/subtasks are as follows:

A – Whether the tweet is offensive (OFF) or non-offensive (NOT).

B – Whether the tweet is targeted (TIN) or untargeted (UNT).

C – If the target is an individual (IND), group (GRP) or other (OTH; e.g., an issue or an organisation).

The dataset does not have an equal number of offensive and non-offensive tweets. Only about one-third of the tweets are marked offensive, to partially account for the fact that most online discourse mainly is non-offensive. The corpus exhibits a larger number of male (∼3000) than female pronouns (∼2500), but is reasonably balanced.

Noticeably, the annotators were very conservative in their classification of tweets as non-offensive. It is unclear whether this was due to a more strict definition provided by the task organisers. For example, it is not immediately clear why tweets such as:[1] *"@USER Ouch!" (23159)*, *"@USER He is a beast" (50771)*, and *"@USER That shit weird! Lol" (31404)* were annotated as offensive.

The annotators furthermore seemed to disagree over the cathartic and emphatic use of swearing, as in *"@USER Oh my Carmen. He is SO FRICK-*

[1]In the examples, tweet IDs are given in parenthesis.

ING CUTE" (39021), *"@USER GIVE ME A FUCKING MIC" (60566)*, and *"@USER why are you so fucking good." (80097)*. These tweets do not really seem to be offensive except for them containing varying degrees of profanity. However, this is inconsistent, with some other tweets annotated not offensive, as expected: *"@USER No fucking way he said this!" (47427)*, and *"@USER IT'S FUCKING TIME!!" (59465)*, although most tweets that contained profanity were included in the offensive class.

Another thing to note is a large amount of political criticism within the tweets in the corpus. Whether it be left wing or right wing, extreme cases seem to be correctly annotated as offensive, while a healthy amount of criticism and political discourse correctly is annotated as non-offensive. The dataset also exhibits a dearth of racist tweets.

4 Methodology

Initially, a suite of features was composed based on those used successfully in previous work such as Waseem and Hovy (2016), Davidson et al. (2017), Nobata et al. (2016) and Burnap and Williams (2015): surface-level token unigrams, bigrams, and trigrams, weighted by TF-IDF; POS tags obtained through the `CMU tagger`[2] (Gimpel et al., 2011), which was specifically developed for the language used on Twitter; sentiment score assigned using a pre-trained model included in `TextBlob`[3]; and count features for URLs, mentions, hashtags, punctuation marks, words, syllables, and sentences.

`Scikit-learn`[4] (Pedregosa et al., 2011) was used as the primary library for modelling and training. L1-regularised Logistic Regression and a Linear Support Vector Classifier stood out initially as the best models. Further experimentation displayed that while those two models exhibited the highest accuracy, their recall of offensive tweets in subtask A and of untargeted insults in subtask B were lower than other classifiers provided in the Scikit-learn library, such as the Passive-Aggressive (PA) classifier (Crammer et al., 2006) and stochastic gradient descent (SGD).

Further exploration showed that the classifiers were not in agreement on certain tweets. This led to the idea of a vote-based ensemble model built on the following five classifiers combined by plurality voting (Kuncheva, 2004): L1-regularised Logistic Regression, L2-regularised Logistic Regression, Linear SVC, SGD, and PA. The ensemble model exhibited the best results in subtasks A and B. In subtask C, the multi-class classification problem and a severe reduction of the size of the training set led to much lower macro-averaged F_1-scores, with the ensemble model performing badly. A deep learning approach, based on an LSTM architecture (Hochreiter and Schmidhuber, 1997), was adopted specifically for this subtask. The model used a 200 dimensional `GloVe embedding`[5] pre-trained on 2 billion tweets (Pennington et al., 2014), with trainability set to False. The embedding layer was followed by a 1D convolution layer with 64 output filters and a Rectified Linear Unit (ReLU) activation function. The output of this layer was down-sampled using a max pooling layer of size 4. These inputs were fed into an LSTM layer of 200 units and subsequently a dense layer of 3 units with a softmax activation function. The model used the 'Adam' optimiser and the categorical cross entropy loss function. Due to the less amount of data, overfitting was quite common on as few as 3 epochs. Therefore, the model benefited from larger dropout values (up to 0.5). This model exhibited a better result than the ensemble model in subtask C, although only by a small margin.

5 Results

The experiments were run in three stages. First, before choosing the models, a mini ablation study was carried out on how various features affected the accuracy and F_1-score metrics of different models. The selected models were then optimised on the training set, before being evaluated on the test dataset.

5.1 Feature Engineering

The initial ablation study was carried out on a small sample space of models: the Linear SVC and L1/L2-penalised Logistic Regression. The results are represented in Table 1.

The ablation analysis revealed that surface-level token/character n-grams are by far the most predictive of the features. An interesting observation is the significantly improved recall of offensive tweets when character n-grams are included.

[2]www.cs.cmu.edu/~ark/TweetNLP/
[3]textblob.readthedocs.io/en/dev/
[4]scikit-learn.org/stable/
[5]nlp.stanford.edu/projects/glove/

Features	Linear SVC			Logistic Regression L1			Logistic Regression L2		
	Acc	F_1	Rec	Acc	F_1	Rec	Acc	F_1	Rec
(1,3) word n-gram	.7694	.7257	.4884	.7671	**.7309**	.5938	.7583	.7193	.5684
+ POS tags	.7647	.7192	.4650	.7584	.7155	.5659	.7482	.7072	.5502
+ Sentiment Score	.7635	.7188	.4920	.7399	.7057	.5997	.7261	.6894	.5752
+ Sentiment Score - POS Tags	.7694	.7265	.5068	.7558	.7226	.6161	.7384	.7033	.5929
+ Count Features	.7673	.7234	.4925	.7422	.7096	.6125	.7327	.7004	.6075
+ Count Features - POS Tags	**.7710**	.7286	.5115	.7587	.7260	.6209	.7478	.7149	.6138
(2,5) char n-gram	.7532	.7185	.6032	.7376	.7080	.6288	.7487	.7203	.6459
+ Sentiment Score	.7539	.7189	.6015	.7315	.7060	.6490	.7503	.7240	.6604
+ Count Features	.7534	.7191	.6068	.7323	.7077	.6557	.7523	.7262	**.6636**

Table 1: Ablation analysis on subtask A, with the training set.

	Subtask A				Subtask B			Subtask C				
	All NOT	All OFF	Linear SVC	Ensemble model	All TIN	All UNT	Ensemble model	All GRP	All IND	All OTH	Ensemble model	LSTM network
F_1	.4189	.2182	.7369	**.7434**	.4702	.1011	**.7079**	.1787	.2130	.0941	.4854	**.5056**
Acc.	.7209	.2790	.8012	**.8023**	**.8875**	.1125	.8833	.3662	.4695	.1643	.6291	**.6385**

Table 2: Test set results (macro-F_1 and accuracy) for all subtasks, with class baselines ("All X").

However, the best F_1-score/accuracy was never achieved with the character n-gram model, and hence only token n-grams were included on the final feature list. Other features provided only small improvements in accordance with previous observations (Wiegand et al., 2018). The addition of POS information seems to cause a reduction in performance, so this feature was dropped, except for in subtask C, where a small positive effect could be observed. Furthermore, artificially balancing the classes by modifying class weights helped alleviate the low recall issue to some extent.

5.2 Training Set

A 10-fold cross-validation was performed on each model used in the ensemble, with the metrics obtained in each fold averaged to obtain a median for each model's performance on the dataset. These initial results were obtained only for subtask A, to decide which models would be a part of the ensemble. Most models used in the ensemble exhibited similar accuracy, but varied in the recall of offensive tweets. It was also observed that models with the higher recall of offensive tweets exhibited equivalently lower recall of non-Offensive tweets. These observations are graphically represented in Figure 1. Small improvements in F_1-score and accuracy were achieved while using the **ensemble model (F_1-score: .7338 and Accuracy: .7720)** over any other single classifier model.

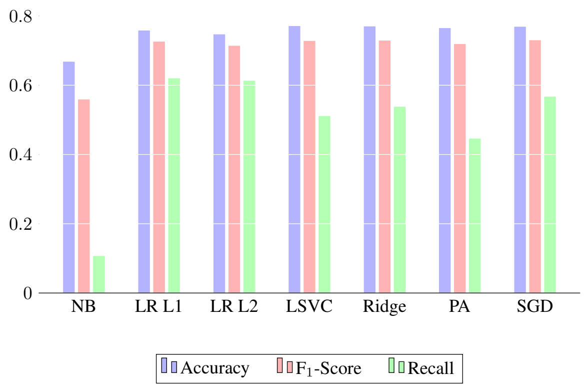


Figure 1: Performance of individual classifiers

5.3 Test Set

After the models were trained, their performance was measured on a separate set of 860 unseen tweets. All F_1-scores provided by the OffensEval organising team were macro-averaged. Baselines for each metric were also provided.

Subtask A: The best single model, Linear SVC came in at .7369 F_1-score and .8012 accuracy, while the ensemble model achieved a slightly improved .7434 F_1 and .8023 accuracy, as highlighted in Table 2. Most models used in the ensemble exhibited similar F_1 and accuracy, but recall of offensive tweets varying in the 0.4–0.7 range, with models with high offensive recall exhibiting equivalent decrease in recall of non-offensive tweets. On the unseen test data, the ensemble model reached a .5792 recall on offensive tweets and .8887 on non-offensive tweets.

Subtask B: This subtask represented a highly imbalanced dataset, with the number of targeted instances (213) dwarfing the number of untargeted instances (27). Here the ensemble model performed the best by far, while the different individual models exhibited high disparity on separation into the two classes. Though the ensemble at .7079 exhibited the highest F_1-score, its accuracy still trailed behind the baseline targeted (TIN) accuracy by a small margin (.8833 vs .8875). The recall of the targeted and untargeted (UNT) tweets were .9343 and .4815, respectively.

Subtask C: Subtask C entailed multi-class classification over the target type of insult. This was the only subtask which exhibited improvement through the inclusion of POS data. As seen in Table 2, the ensemble model achieved .4854 F_1 and .6291 accuracy. The LSTM network provided better results, coming in at .5056 F_1-score and .6385 accuracy, when using a 200-dimensional GloVe embedding. In this subtask, as expected, classification of the minority class, OTH, proved to be the most troublesome. Both the ensemble model and the LSTM exhibited very low recall on that class: .0571 and .0857, respectively. The recall of the IND and GRP classes were .7800 and .6923, respectively.

6 Error Analysis

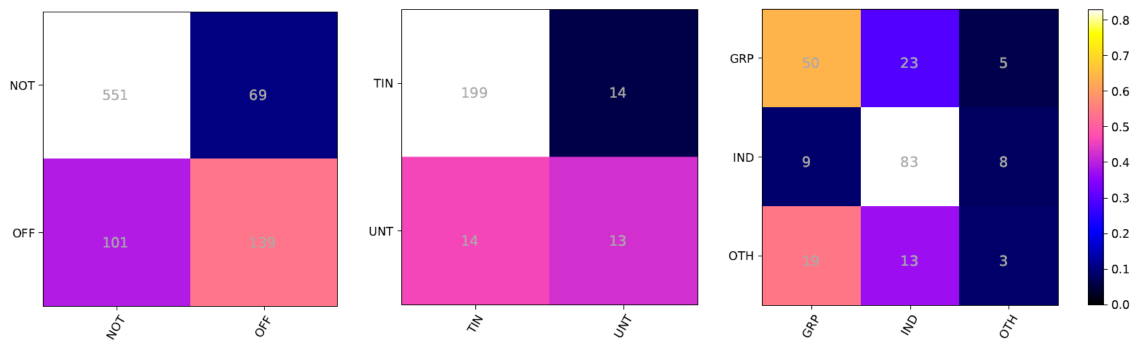
This section gives a short qualitative analysis of the misclassifications in each subtask and hypothesises potential reasons for the errors.

Subtask A: As seen in Figure 2a, the ensemble model had more difficulty identifying offensive tweets than the non-offensive ones. As also noted in previous work by Davidson et al. (2017) and others, we see that the classifier finds it difficult to identify offensive tweets that lack profanity such as *"@USER Get back on your peanut farm old man" (24726)* and *"@USER She is such a witch. All she needs is a broom" (49813)*. The classifier also faced issues in classifying political discourse, as it may have learned trends of words such as 'MAGA', 'Trump', 'Liberals' and 'Conservatives' being appearing relatively often under the OFF (offensive) label. This leads to misclassification of tweets such as *"@USER Up next: liberals calling us out for calling him guilty." (59807)* and *"@USER there is a point where even liberals must question motives" (15788)* as offensive.

Subtask B: Due to the highly imbalanced data set, the minority class (UNT) as expected accounted for most of the misclassifications, as seen in Figure 2b. The simple trend deduced was that tweets with pronouns such as 'she', 'your', 'he', and 'I' were biased to be classified as targeted (TIN). This leads to misclassification of untargeted insults such as, *"@USER @USER Still no excuse... Where TF are her parents??? They are using him &amp; he is using her" (10641)* and *"@USER If someone is being too nice to you at happy hour and asking probing questions about what you do at Pub Citizen....make sure to troll them and say you're with Antifa or something." (58699)*, *"@USER I hate him im so fucking sorry" (91969)*. The opposite is also true, with targeted insults that contain no pronouns being misclassified as untargeted: *"@USER Google go to hell!" (52798), "@USER and bale is shit" (47806).*

Subtask C: Most misclassification in this subtask occurred on the Other (OTH) label; see Figure 2c. Here most tweets labelled OTH were classified under the Group (GRP) class, due to close similarity between the two labels. Consider the following examples: *"@USER @USER Because 45% of Americans are too lazy to vote. Non-voters skew liberal. And too many liberals who do vote throw their vote away on 3rd party losers. Next question?" (82171)* and *"@USER @USER @USER @USER Connections are vital with all of the crap Twitter forces on conservatives." (89193)*. Both these tweets are classified as GRP insults, probably due to the presence of terms such as 'Americans', 'liberals', and 'conservatives' that tend to relate to groups, while actually being annotated as OTH as they address an issue rather than a group.

There were also a considerable number of misclassifications of OTH class tweets as IND. These misclassifications are justified on similar grounds. Examples include *"@USER Google go to hell!" (52798)*, and *"@USER get your shit together" (18315)*.



(a) Sub-task A: Ensemble model (b) Sub-task B: Ensemble model (c) Sub-task C: LSTM network

Figure 2: Confusion matrices (X-axis = predicted label; Y-axis = true label)

7 Conclusion

The idea of a hierarchical classification of offensive language is a step in the right direction in reducing the ambiguity existing between various similar subtasks. It is yet to be seen, however, how effective this method would be in synthesising more specific subsets of offensive language. For example, the cyberbullying subtask instances may yield either OFF, TIN or IND labels at each level of classification, but we are unaware of how effectively models developed for the OffensEval subtask perform on cyberbullying data sets. Some issues that have plagued offensive language detection — such as the problem of ambiguity and overlap between various subtasks — could effectively be solved if the idea of hierarchical classification achieves what it sets out to do.

Consistent with previous work, we find that it is difficult to classify non-offensive tweets containing profanity and offensive tweets lacking profanity. We also found that a similar issue persists with tweets that are politically motivated and valid criticism incorrectly classified as offensive and similarly, political hate incorrectly classified as non-offensive.

On the topic of selecting a classification model, it is noteworthy that even a simple and crude deep learning model such as the one used here can obtain better results than a more polished ensemble model. Except for surface level n-grams, most features are not as predictive as we would like them to be.

The data analysis showed that even though the annotators of the OLID data set were experienced with the platform, there still exist quite a few cases of erroneous classification by the annotators, just as noted for other datasets (Waseem and Hovy, 2016; Davidson et al., 2017; Nobata et al., 2016), for which amateur annotators were found unreliable.

Offensive language detection has proven to be a more layered issue than was initially expected, but with various developments in research the task seems surmountable. Future work must focus on building upon previous endeavours, to reduce the redundancy between subtasks and publications. The OffensEval shared task is a significant step forward in achieving this goal and we look forward to seeing how future research will be affected by the work that has been done here.

References

Peter Burnap and Matthew Leighton Williams. 2015. Hate speech, machine classification and statistical modelling of information flows on Twitter: Interpretation and communication for policy decision making. *Policy and Internet*, 7(2):223–242.

Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. 2012. Detecting offensive language in social media to protect adolescent online safety. In *Proceedings of the 2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust*, pages 71–80, Amsterdam, Netherlands. IEEE.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Maral Dadvar, Dolf Trieschnigg, Roeland Ordelman, and Franciska de Jong. 2013. Improving cyberbullying detection with user context. In *Advances in Information Retrieval*, pages 693–696. Springer.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In

Proceedings of the 11th International Conference on Web and Social Media, pages 512–516, Montréal, Québec, Canada. AAAI Press.

Karthik Dinakar, Roi Reichart, and Henry Lieberman. 2011. Modeling the detection of textual cyberbullying. In *The Social Mobile Web*, pages 11–17.

Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large scale crowdsourcing and characterization of Twitter abusive behavior. *CoRR*, abs/1802.00393.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using convolutional neural networks to classify hate-speech. In (Waseem et al., 2017a), pages 85–90.

Lei Gao and Ruihong Huang. 2017. Detecting online hate speech using context aware models. *CoRR*, abs/1710.07395.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for Twitter: annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 2, short papers, pages 42–47, Portland, Oregon, USA. ACL.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Raquel Justo, Thomas Corcoran, Stephanie M. Lukin, Marilyn Walker, and M. Inés Torres. 2014. Extracting relevant knowledge for the detection of sarcasm and nastiness in the social web. *Knowledge-Based Systems*, 69(1):124–133.

Ludmila I. Kuncheva. 2004. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, New York, New York, USA.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in discriminating profanity from hate speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Yashar Mehdad and Joel R. Tetreault. 2016. Do characters abuse more than words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303, Los Angeles, California, USA. ACL/SIGDIAL.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*, pages 145–153, Montréal, Québec, Canada. IW3C2.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar. ACL.

Georgios Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Effective hate-speech detection in twitter data using recurrent neural networks. *Applied Intelligence*, 48(12):47304742.

Anna Schmidt and Michael Wiegland. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the 5th International Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia, Spain. ACL.

Elise Fehn Unsvåg and Björn Gambäck. 2018. The effects of user features on Twitter hate speech detection. In *Proceedings of the 2nd Workshop on Abusive Language Online*, pages 75–86, Brussels, Belgium. ACL.

Cynthia Van Hee, Els Lefever, Ben Verhoeven, Julie Mennes, Bart Desmet, Guy De Pauw, Walter Daelemans, and Veronique Hoste. 2015. Detection and fine-grained classification of cyberbullying events. In *Proceedings of Recent Advances in Natural Language Processing, Proceedings*, pages 672–680.

Zeerak Waseem, Wendy Hui Kyong Chung, Dirk Hovy, and Joel Tetreault, editors. 2017a. *Proceedings of the First Workshop on Abusive Language Online*. ACL, Vancouver, Canada.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017b. Understanding abuse: A typology of abusive language detection subtasks. In (Waseem et al., 2017a), pages 78–84.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter. In *Proceedings of the North American Chapter of the Association for Computational Linguistics, Student Research Workshop*, pages 88–93, San Diego, California, USA. ACL.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 shared task on the identification of offensive language. In *Proceedings of the GermEval 2018 Workshop*, Austrian Academy of Sciences, Vienna, Austria.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399, Perth, Australia. IW3C2.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from bullying traces in social media. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 656–666, Montréal, Canada. ACL.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the type and target of offensive posts in social media. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Minneapolis, Minnesota, USA. ACL.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval)*. ACL.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting hate speech on Twitter using a convolution-GRU based deep neural network. In *Proceedings of the 15th International Semantic Web Conference*, pages 745–760, Heraklion, Greece. Springer Verlag.

NLP@UIOWA at SemEval-2019 Task 6: Classifying the Crass using Multi-windowed CNNs

Jonathan Rusert and **Padmini Srinivasan**
Department of Computer Science
University of Iowa
Iowa City, IA, USA
{jonathan-rusert, padmini-srinivasan}@uiowa.edu

Abstract

This paper proposes a system for OffensEval (SemEval 2019 Task 6), which calls for a system to classify offensive language into several categories. Our system is a text based CNN, which learns only from the provided training data. Our system achieves 80 - 90% accuracy for the binary classification problems (offensive vs not offensive and targeted vs untargeted) and 63% accuracy for trinary classification (group vs individual vs other).

1 Introduction

Background. Social media (e.g. Twitter, Facebook) is widely used today. For example, 68% of all Americans report owning a Facebook account in 2018 (Smith and Anderson, 2018), while 71% of Americans (within the ages of 18-24) report using Twitter. Online gaming is a second popular use of the internet, reaching upwards of 80 - 100 million monthly users depending on game (Goslin, 2018). These uses demonstrate the internet as a way for humans to connect with others. However, connecting with others can carry a downside. More than 1 in 3 young people have been cyberbullied online (cyb, 2018), this extends to around half of teens. Besides cyberbullying, offensive language, on a public forum, can cause users to stay away from certain platforms. Because of this, companies have increased their efforts to remove offensive language from their platforms (Terdiman, 2018). With the large amount of traffic these platforms see, a purely manual approach to detecting/removing offensive language is impossible, which means an automated approach is needed to help. OffensEval (Zampieri et al., 2019b) provides a community driven opportunity to build such systems. We approach this problem of classifying offensive tweets with a CNN architecture trained on the provided training dataset.

2 Proposed Approach

Our system is a variation of a Convolutional Neural Network (CNN), which was chosen since it has seen success previously with classification tasks

CNN Infrastructure. We experimented with two different Convolutional Neural Networks. The first whose architecture is based on the CNN originally proposed in (Kim, 2014) (CNN 1), and the second a combination of multiple CNNs (CNN 2). We further discuss each of these CNNs and their comparison on the training data. The visual structure for CNN 1 and CNN 2 can be found in figure 1 and figure 2 respectively.

Preprocessing. Both CNN 1 and CNN 2 begin by preprocessing the text of the tweet. As noted in section 3, URLs and user mentions are already denoted as URL and USER. Basic cleaning of the text is applied, includes removal of punctuation, converting text to lowercase, and filtering of stopwords via NLTK's stopword list[1]. Finally, all separated words are tokenized via nltk's tokenize() function[2].

Embedding Layer. CNN 1 and CNN 2 encode the text of a tweet as a word embedding with dimension j. We experimented with several j arriving at j = 100. Word embeddings for Non-Out of Vocabulary (OOV) words are obtained from Glove (Pennington et al., 2014) which has been trained on Twitter data[3]. Experiments were also conducted with Glove common crawl data, but no visible improvement was found. OOV words are randomly initialized as a word embedding. The embedding layer takes in i word embeddings of length j, where the i word embeddings are combined in the same sequential order as they appear in the tweet. We choose i as the length of the

[1]nltk.org/api/nltk.corpus.html
[2]nltk.org/api/nltk.tokenize.html
[3]nlp.stanford.edu/projects/glove/

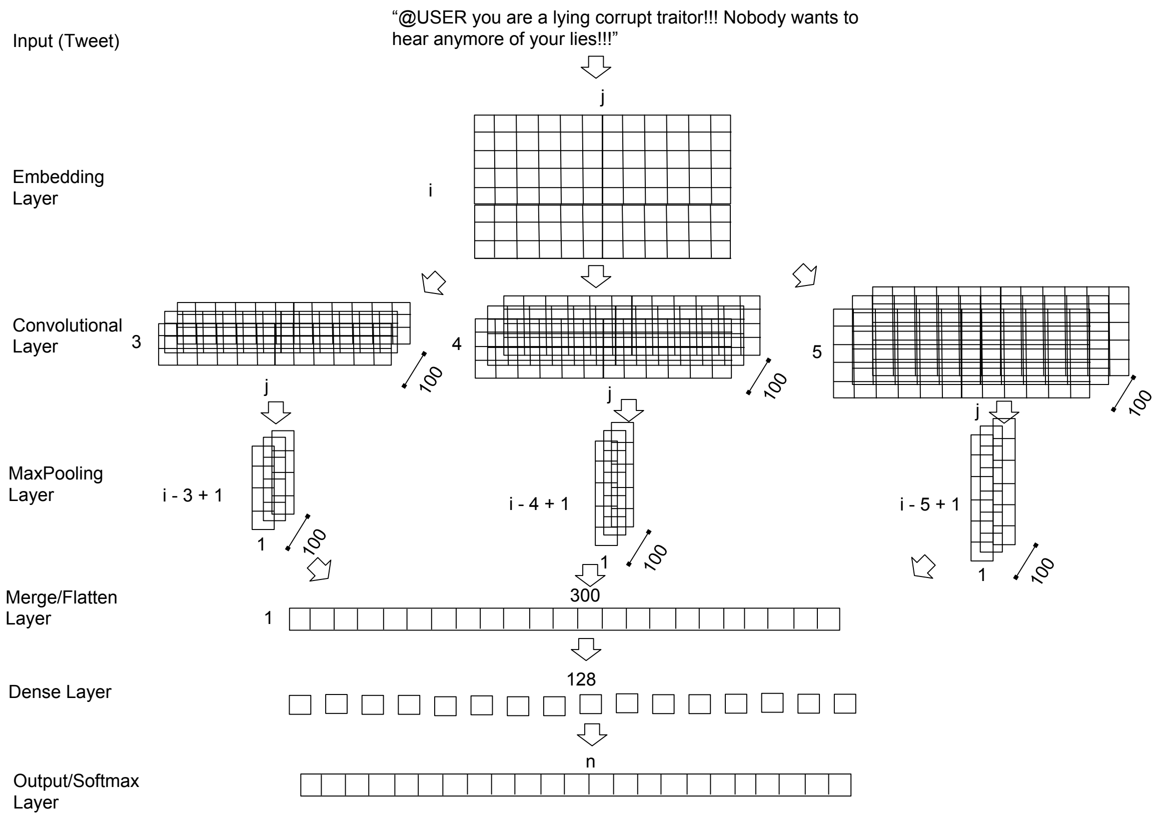


Figure 1: CNN 1 's Architecture

longest tweet (i.e. number of words after preprocessing). Any tweets less than *i* length are padded with zero embeddings at the end. CNN 1 and CNN 2 differ at this point and will be examined separately.

CNN 1 **Convolutional Layer.** CNN 1 applies three $k \times j$ convolutional windows to the embedding layer: a 3 x *j*, a 4 x *j*, and a 5 x *j* window. Applying each window to the embedding layer results in a $(i - k + 1) \times 1$ output, where k = {3,4,5} and corresponds to the length of the window. 100 filters of each window are applied to the embedding layer resulting in 100 $(i - k + 1) \times 1$ outputs for each *k*.

CNN 1 **Max pooling/Merge Layer** A max pooling of size $(i - k + 1) \times 1$ is applied to each separate filter output from the convolutional layer. The resulting outputs from all three max pooling streams are merged together then flattened to a 300 neuron layer.

CNN 1 **Dense Layer/Output Layer** The flattened layer is fed into a dense layer consisting of 128 neurons. ReLu is chosen as the activation function. Finally, the output of the dense layer is passed to the output layer of size *n* where *n* = number of classes. The output layer uses a softmax function as activation.

CNN 2 **Convolutional Layer.** CNN 2 applies three sets of three convolution windows to the embedding layer, each window in the format $k \times j$. The first set of convolution windows are $k = [2, 3, 4]$, the second are $k = [3, 4, 5]$, and third are $k = [4, 5, 6]$. Similar to CNN 1 , applying these filters results in a $(i - k + 1) \times 1$ output, and 100 filters exist for each *k* for each set of windows.

CNN 2 **Max pooling/Merge Layer.** Max pooling, in this instance, behaves similarly to CNN 1 . However, instead of merging all the pooled layers, only those in the same set are merged. This results in three separate flattened 300 neuron layers.

CNN 2 **Separate Dense layers** The three separate merged layers are fed through two dense layers consisting of 128 neurons each. ReLu activation function is used for all dense layers. Each merged layer is fed to their own respective dense layers. The second dense layers are finally merged together.

CNN 2 **Final Dense/Output Layers** The merged layer is fed through two more 128 dense layers with ReLu activation. Finally, the result is

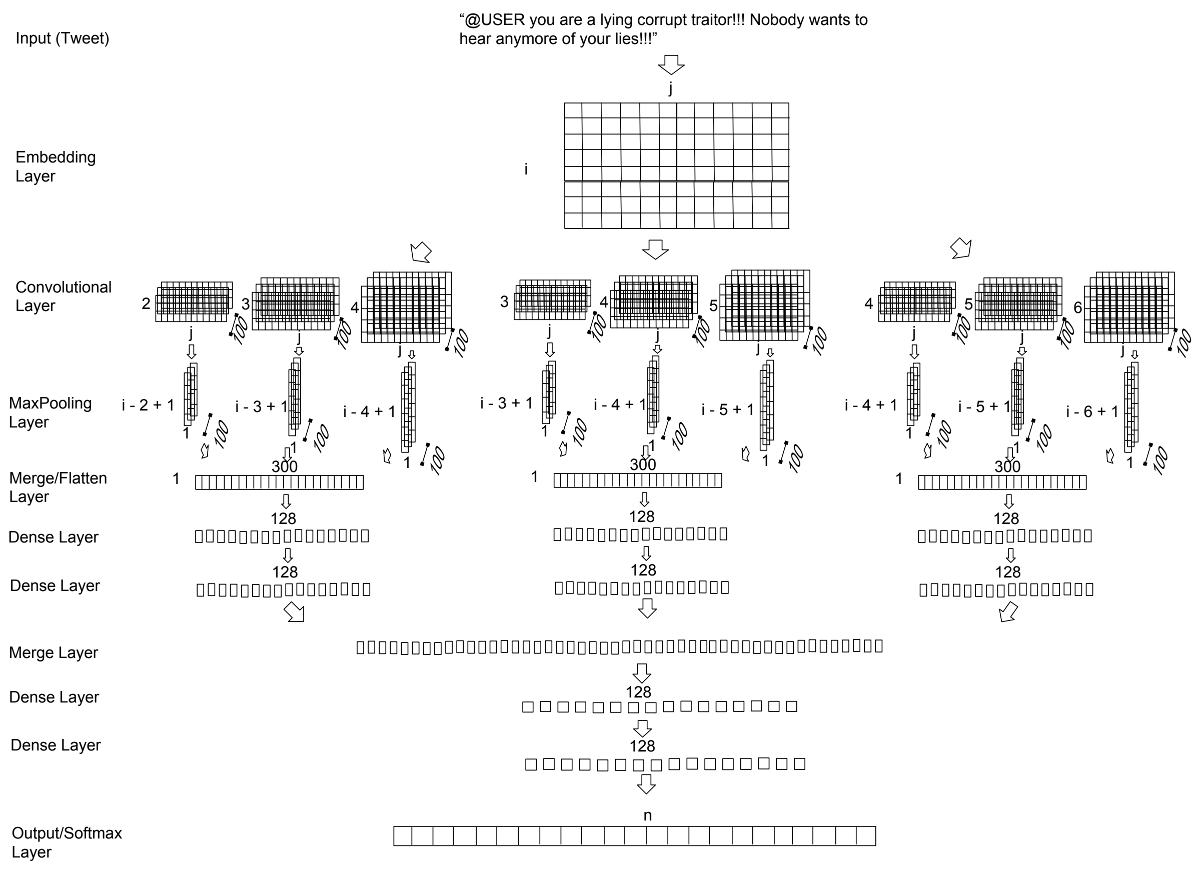


Figure 2: CNN 2 's Architecture

fed to the output layer of size n with softmax activation.

Hyperparameters/Training We experimented with different epochs and batch sizes and ended up finding epochs=30 and batch size = 50 worked best for our models. The only data trained on was the training data provided. More on this data in section 3. The system was implemented with Keras[4] and Tensorflow as the backend.

3 Dataset

Training Set. The data collection methods used to compile the dataset provided in OffensEval is described in Zampieri et al. (2019a). The training data set provided consists of 13,240 tweets. Each tweet, consists of up to three classifications, which correspond to subtasks further described in section 4. The classifications are as follows:

i. OFF - This post contains offensive language or a targeted (veiled or direct) offense

ii. NOT - This post does not contain offense or profanity.

iii. TIN - A post containing an insult or threat to an individual, a group, or others

iv. UNT - A post containing non-targeted profanity and swearing.

v. IND - The target of the offensive post is an individual: a famous person, a named individual or an unnamed person interacting in the conversation.

vi. GRP - The target of the offensive post is a group of people considered as a unity due to the same ethnicity, gender or sexual orientation, political affiliation, religious belief, or something else.

vii. OTH - The target of the offensive post does not belong to any of the previous two categories (e.g., an organization, a situation, an event, or an issue)

A tweet which is classified as OFF, can be further classified into TIN or UNT. If classified as TIN, the tweet can be further classified into IND, GRP, or OTH. A breakdown of the frequency of each class label can be found in table 1.

[4]https://keras.io/

OFF				NOT
4400				8840
TIN			UNT	
3876			524	
IND	GRP	OTH		
2407	1074	395		

Table 1: A breakdown of frequency of labels of tweets, the classes underneath are further classifications of classes above (e.g. a tweet labeled IND is also labeled TIN and OFF)

Test Set. The test set provided follows the same classification rules as training and consists of 860 tweets. The 860 tweets can be classified into OFF or NOT, then 240 OFF tweets can be classified as TIN or UNT, and finally 213 TIN tweets can be classified as IND, GRP or OTH.

4 Subtasks

OffensEval divided the overall task of identifying/classifying offensive language into three subtasks, subtask A, B, and C.

Subtask A. Subtask A requires a system to classify tweets as either offensive (OFF) or not offensive (NOT). An example of a tweet marked as OFF (in provided training):

@USER you are a lying corrupt traitor!!! Nobody wants to hear anymore of your lies!!!.

An example of a tweet marked as NOT:

@USER Buy more icecream!!!.

A more expanded look at the training data can be found in section 3.

Subtask A Results. As our system only trained on the provided gold standard, this data set was used to gauge the effectiveness of our two systems. Five fold cross validation was used for predicting training data. The results for subtask A on training data can be found in table 2. CNN 1 and CNN 2 achieve similar results, an accuracy of 0.7468 and 0.7555, and a macro F1 score of 0.7130 and 0.7114, respectively. The results for our systems' performance on OffensEval test data subtask A can be found in table 3. As with the training data, CNN 1 and CNN 2 perform similarly on this task, with CNN 1 achieving 0.8 accuracy and a macro F1 score of 0.73.

Subtask B. Subtask B requires further classification of OFF tagged into two categories, targeted (TIN) and untargeted (UNT). An example of untargeted tweet is:

System	Acc.	Pr.	Re.	F1
CNN 1	0.7469	0.7145	0.7117	0.7130
CNN 2	0.7555	0.7256	0.7036	0.7114

Table 2: Subtask A Training Data Results, Pr.=Macro Precision, Re.=Macro Recall, F1=Macro F1 Score

System	Acc.	Pr.	Re.	F1
CNN 1	0.7988	0.7552	0.7175	0.7314
CNN 2	0.7767	0.7250	0.7379	0.7306
All NOT	0.7209			0.4189
All OFF	0.2790			0.2182

Table 3: Subtask A Test Results, Pr.=Macro Precision, Re.=Macro Recall, F1=Macro F1 Score. All OFF, NOT are baselines where that specific label was assigned to all tweets.

@USER @USER My favourite part of this is watching all the conservatives lose their minds as usual. Once again the Democrats a re being mean to us boo-hoo. LOL.

An example of targeted:

@USER You are a complete knob! It's ppl like you who are messing up this country. More details on data in section 3.

Subtask B Results. The results for cross validation on subtask B's training data are found in table 4. CNN 2 achieves a greater accuracy over CNN 1 on this subtask, 0.8723 compared to 0.8222, but still achieves a smaller macro F1 score of 0.5673 compared to 0.5827. Subtask B test results for our systems are found in table 5. Similar to training, CNN 2 outperforms CNN 1 in accuracy, 0.8958 to 0.8750, but achieves a similar trend in macro F1 scores, 0.6511 and 0.6528.

Subtask C. Subtask C requires further classification of those tweets tagged are targeted (TIN), into three classes, Individual (IND), Group (GRP), and Other (OTH). Examples:

IND tweet -*@USER You are a complete knob! It's ppl like you who are messing up this country*

GRP tweet - *@USER Assuming liberals are unarmed would be a grave mistake by the deplorables.*

System	Acc.	Pr.	Re.	F1
CNN 1	0.8222	0.5815	0.5839	0.5827
CNN 2	0.8723	0.6442	0.5545	0.5673

Table 4: Subtask B Training Data Results, Pr.=Macro Precision, Re.=Macro Recall, F1=Macro F1 Score

System	Acc.	Pr.	Re.	F1
CNN 1	0.8750	0.6732	0.6385	0.6528
CNN 2	0.8958	0.7478	0.6179	0.6511
All TIN	0.8875			0.4702
All UNT	0.1125			0.1011

Table 5: Subtask B Test Results, Pr.=Macro Precision, Re.=Macro Recall, F1=Macro F1 Score. All TIN, UNT are baselines where that specific label was assigned to all tweets.

System	Acc.	Pr.	Re.	F1
CNN 1	0.6925	0.5372	0.5224	0.5282
CNN 2	0.6772	0.5147	0.5136	0.5140

Table 6: Subtask C Training Data Results, Pr.=Macro Precision, Re.=Macro Recall, F1=Macro F1 Score

OTH tweet - *@USER Shooting in USA is so common no one is talking about gun control any more.*

More details on subtask C data in section 3.

Subtask C Results. The results for subtask C's cross validation on training data can be found in table 6. CNN 1 slightly outperforms CNN 2 in this task, achieving an accuracy of 0.6925 over 0.6772 and a macro F1 score of 0.5282 over 0.5140. Subtask C's test data results can be found in table 7. As subtask C is a three class problem (compared to the two class problem of A, B), the accuracy and macro F1 scores are lower overall. As with training, CNN 1 slightly outperforms CNN 2 in both accuracy, 0.6291 to 0.6197, and macro F1 score, 0.5061 to 0.4939.

5 Discussion

Systems outperform single labels. On test data, for all three subtasks, our system outperforms the baseline, provided by organizers, for assigning a single label for all tweets. Outperforming the best baseline (labeling tweets with highest frequent label) in terms of F1 by 0.32 in subtask A, 0.18 in subtask B, and 0.29 in subtask C. Outperformances in accuracy are seen in all three subtasks as well.

System	Acc.	Pr.	Re.	F1
CNN 1	0.6291	0.5513	0.5148	0.5061
CNN 2	0.6197	0.5079	0.5014	0.4939
All GRP	0.3662			0.1787
All IND	0.4695			0.2130
All OTH	0.1643			0.0941

Table 7: Subtask C Test Results, Pr.=Macro Precision, Re.=Macro Recall, F1=Macro F1 Score. All GRP, IND, OTH are baselines where that specific label was assigned to all tweets.

		NOT	OFF	
Key	NOT	559	61	620
	OFF	112	128	240
		System		860

Table 8: CNN 1 's Confusion matrix for subtask A

		TIN	UNT	
Key	TIN	201	12	213
	UNT	18	9	27
		System		240

Table 9: CNN 1 's Confusion matrix for subtask B

Results follow data distribution. The test confusion matrices for the higher scoring system (CNN 1), for subtask A, B, and C, can be found in table 8, table 9, and table 10 respectively. Training data for OFF and NOT make up 67% and 33% respectively. For test data, the percentages are 28% for OFF and 78% for NOT. As expected, our system identifies better identifies NOT (559/620 tweets) compared to OFF (128/240 tweets). Similar results occur for subtask B, TIN (201/213) compared to UNT(9/27), and subtask C, GRP (33/78) compared to (95/100) compared to (6/35). These results all follow distribution of training data, which might point to lack of training data for poorer results for smaller classes since deep learning systems depend on a good amount of training data.

Added complexity of CNN 2 **adds little to no improvement.** Although CNN 2 shows greater performance in accuracy on the training data for subtasks A and B, the performance does not follow through in the test data, as CNN 1 outperformed CNN 2 in subtask A and C for accuracy and all

		GRP	IND	OTH	
	GRP	33	34	11	78
Key	IND	3	95	2	100
	OTH	12	17	6	35
			System		213

Table 10: CNN 1 's Confusion matrix for subtask C

three subtasks for macro F1 score. Although CNN 2 performs slightly worse, this may not be due to the structure itself, CNN 2 is currently trained the traditional way (updates all weights as once) but it may be necessary for the branches to be trained separately. This requires further testing in the future.

6 Related Work

Offensive language detection and classification has become increasingly relevant in recent years with the rise of social media. Subsequently, researchers have also begun to look at aggression, cyberbullying, hate speech, and abusive language identification.

Cyberbullying. Cyberbullying detection has been approach by several teams. Dinakar et al. (2011) show that binary classifiers for individual labels outperforms multi-label classifiers. Xu et al. (2012) demonstrate that social media is a rich environment for studying cyberbulling with NLP. Dadvar et al. (2013) show the effectiveness of including context around a comment.

Abusive Language. Abusive language has also seen increase in study. Nobata et al. (2016) construct a machine learning algorithm and test with different lexical features, outperforming at the time state-of-the-art methods. Mubarak et al. (2017) expand abusive language identification to Arabic social media. Fišer et al. (2017) propose a legal framework, dataset and anotation schema for abusive online language in Slovene. Su et al. (2017) propose a system which can not only detect, but also rephrase abusive language in Chinese. Waseem et al. (2017) propose breaking abusive language identification into further subtasks. Founta et al. (2018) leverage crowd sourcing to produce a large (80,000) annotated data set of abusive Twitter language.

Hate speech. Hate speech identification has come to the forefront for research as it deals with current hot button issues (e.g. racism, sexism). Schmidt and Wiegand (2017) and Fortuna and Nunes (2018) compile a surveys of current hate speech detection.

Other teams have brought to question how we view and handle hate speech. Ross et al. (2016) show the difficulty of annotating hate speech and propose handling classification as non-binary. Malmasi and Zampieri (2017) establish lexical baselines for hate speech detection by applying supervised classification methods and Malmasi and Zampieri (2018) show the problems which can arise when distinguishing profanity from hate speech. ElSherief et al. (2018) further look to understand hate speech by looking into the target of hate speech (i.e. at a individual or more general group).

Machine learning classifiers are leveraged in this field as well. Kwok and Wang (2013) employ a machine learning classifier to identify racist tweets. Burnap and Williams (2015) test a machine learning system on different n-gram features to identify hate speech on Twitter. Tulkens et al. (2016) use hate speech dictionaries along with support vector machines to identify racism on Dutch social media. Schofield and Davidson (2017) demonstrate three standard methods for producing features for text classification, targeting specifically the problem of automatic hate speech identification.

Subsequently, deep learning is seen in hate speech detection as well. Djuric et al. (2015) train and leverage comment embeddings to help identify hate speech. Gambäck and Sikdar (2017) show the effectiveness of convolutional neural networks (CNN) when identifying hate speech. Zhang et al. (2018) identify hate speech using a convolution-GRU based deep neural network.

Offensive Language. Offensive language identification aims to broaden the scope of negative language identification. Wiegand et al. (2018) proposed and ran a GermEval task similar to OffensEval, which had participants classify offensive language as offensive or other, then further classify the offensive tagged language.

7 Conclusion/Future Work

We have proposed and tested two different CNN architectures for identifying offensive language. Future work would aim to improve the current CNN design by testing different word windows and training techniques. Furthermore, since deep learning performs better with large amounts of training data, increasing the training data, perhaps even with silver standards if not gold, should help further improve the system's predictions.

References

2018. Cyber Bullying Statistics.

Pete Burnap and Matthew L Williams. 2015. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet*, 7(2):223–242.

Maral Dadvar, Dolf Trieschnigg, Roeland Ordelman, and Franciska de Jong. 2013. Improving cyberbullying detection with user context. In *Advances in Information Retrieval*, pages 693–696. Springer.

Karthik Dinakar, Roi Reichart, and Henry Lieberman. 2011. Modeling the detection of textual cyberbullying. In *The Social Mobile Web*, pages 11–17.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pages 29–30. International World Wide Web Conferences Steering Committee.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Darja Fišer, Tomaž Erjavec, and Nikola Ljubešić. 2017. Legal Framework, Dataset and Annotation Schema for Socially Unacceptable On-line Discourse Practices in Slovene. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large Scale Crowdsourcing and Characterization of Twitter Abusive Behavior. *arXiv preprint arXiv:1802.00393*.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

A. Goslin. 2018. Fortnite has 78.3 million monthly players, according to Epic.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Irene Kwok and Yuzhou Wang. 2013. Locate the hate: Detecting Tweets Against Blacks. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 467–472.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Hamdy Mubarak, Darwish Kareem, and Magdy Walid. 2017. Abusive Language Detection on Arabic Social Media. In *Proceedings of the Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive Language Detection in Online User Content. In *Proceedings of the 25th International Conference on World Wide Web*, pages 145–153. International World Wide Web Conferences Steering Committee.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Wojatzki. 2016. Measuring the Reliability of Hate Speech Annotations: The Case of the European Refugee Crisis. In *Proceedings of the Workshop on Natural Language Processing for Computer-Mediated Communication (NLP4CMC)*, Bochum, Germany.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Alexandra Schofield and Thomas Davidson. 2017. Identifying Hate Speech in Social Media. *XRDS: Crossroads, The ACM Magazine for Students*, 24(2):56–59.

A. Smith and M. Anderson. 2018. Social Media Use in 2018.

Huei-Po Su, Chen-Jie Huang, Hao-Tsung Chang, and Chuan-Jie Lin. 2017. Rephrasing Profanity in Chinese Text. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

D. Terdiman. 2018. Heres How Facebook Uses AI To Detect Many Kinds Of Bad Content.

Stéphan Tulkens, Lisa Hilte, Elise Lodewyckx, Ben Verhoeven, and Walter Daelemans. 2016. A Dictionary-based Approach to Racism Detection in Dutch Social Media. In *Proceedings of the Workshop Text Analytics for Cybersecurity and Online Safety (TA-COS)*, Portoroz, Slovenia.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval*.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from bullying traces in social media. In *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 656–666. Association for Computational Linguistics.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

NLPR@SRPOL at SemEval-2019 Task 6: Linguistically enhanced deep learning offensive sentence classifier

Alessandro Seganti[1], Helena Sobol[1], Iryna Orlova[1], Hannam Kim[2], Jakub Staniszewski[1], Tymoteusz Krumholc[1], Krystian Koziel[1]
[1] Samsung R&D Institute Poland
[2] Samsung Electronics, Korea
a.seganti@samsung.com

Abstract

The paper presents a system developed for the SemEval-2019 competition Task 5 *hat-Eval* Basile et al. (2019) (team name: *LU Team*) and Task 6 *OffensEval* Zampieri et al. (2019b) (team name: *NLPR@SRPOL*), where we achieved 2^{nd} position in Subtask C. The system combines in an ensemble several models (LSTM, Transformer, OpenAI's GPT, Random forest, SVM) with various embeddings (custom, ELMo, fastText, Universal Encoder) together with additional linguistic features (number of blacklisted words, special characters, etc.). The system works with a multi-tier blacklist and a large corpus of crawled data, annotated for general offensiveness. In the paper we do an extensive analysis of our results and show how the combination of features and embedding affect the performance of the models.

1 Introduction

In 2017 two-thirds of all adults in the United States have experienced some form of online harassment (Duggan, 2017).[1] This, together with various episodes of online harassment, boosted research on the general problem of recognizing and/or filtering offensive language on the Internet. Still, recognizing if a sentence expresses hate speech against immigrants or women, understanding if a sentence is offensive to a group of people, an individual or others – these tasks continue to be very difficult for neural networks and machine learning models to accomplish. In order to do this, various implementations have been proposed; for the most successful recent approaches see Pitsilis et al. (2018); Founta et al. (2018); Wulczyn et al. (2017); Waseem and Hovy (2016); Park and Fung (2017); Davidson et al. (2017). Most of them use various combination of features to recognize these characteristics.

This article presents a system that we have implemented for recognizing if a sentence is offensive. The system was developed for two SemEval-2019 competition tasks: Task 5 *hatEval* "Multilingual detection of hate speech against immigrants and women in Twitter" Basile et al. (2019) (team name: *LU Team*) and Task 6 *OffensEval* "Identifying and categorizing offensive language in social media" (Zampieri et al., 2019b) (team name: *NLPR@SRPOL*). Table 1 shows the results that we achieved with our system in the SemEval-2019 competitions.

Competition	Placement
Task 6-A	8th position
Task 6-B	9th position
Task 6-C	2nd position
Task 5-A	8th position (ex aequo)

Table 1: SemEval-2019 results.

In order to create a highly accurate classifier, we combined state-of-the-art AI with linguistic findings on the pragmatic category of impoliteness (Culpeper, 2011; Jay and Janschewitz, 2008; Brown and Levinson, 1987). We achieved this by deciding on the factors that point to the impoliteness of a given expression (for the blacklists) or the entire sentence (for corpus annotation). Such factors led us to divide the blacklist into "offensive" and "offensive in context", as most linguistic studies of impoliteness focus on various aspects of the context. Furthermore, linguistic research made it possible to arrive at a maximally general definition of offensiveness for the crowdsourced annotators.

[1] Due to the topic of the SemEval-2019 Tasks 5 and 6, the present paper contains offensive expressions spelled out in full. These are solely illustrations of the problems under consideration. They should not be interpreted as expressing our views in any way.

The article is organized as follows. Section 2 presents the current state of the art for offensive sentence classification. Section 3 explains the architecture of our system (features, models and ensembles). Section 4 describes the datasets and how they were created. Section 5 shows the results of the SemEval-2019 tasks in detail, motivating which combination of features and models was the best. Finally, section 6 offers conclusions together with our plans for future research.

2 Related work

In recent years, the problem of recognizing if a sentence is offensive or not has become an important topic in the machine learning literature. The problem itself has different declinations depending on the point of view. Currently there are three main areas of research in this topic in the machine/deep learning community:

1. Distinguishing offensive language from non-offensive language;
2. Solving biases in deep learning systems;
3. Recognizing more specific forms of offensivness (e.g. racism, sexism etc.).

The main problem with each of the tasks is the amount of data available to researchers for experimenting with their systems. This – together with the fact that it is difficult to clearly define what is offensive/racist/sexist or not – makes the three problems listed above very difficult for a deep learning system to solve.

Articles have showed that there is a strong bias in text and embeddings, and have tried to solve this bias using different techniques (Zhao et al., 2018; Dixon et al.; Bolukbasi et al., 2016). Furthermore, thanks to a dataset defined in Waseem and Hovy (2016) and Waseem (2016), various works have gone in the direction of recognizing sexism and racism in tweets (Pitsilis et al., 2018; Park and Fung, 2017).

Another field of work was recognizing offensiveness in the Wikipedia internal discussion forum dataset (Wulczyn et al., 2017). This dataset has led to other articles making systems for distinguishing between offensive and non-offensive language (Founta et al., 2018; Pitsilis et al., 2018; Kumar et al., 2018; Gröndahl et al., 2018; Li, 2018; Park and Fung, 2017; Aken et al., 2018).

Linguistic expertise enhanced the functionality at two stages: sentence annotation (described in detail in Section 4) and active creation of blacklists (Section 3). The completion of these tasks breaks new ground, as there exist no corpus linguistic studies on the generality of offensive language, to the best of our knowledge. Recent approaches of narrower scope are Dewaele (2015) and McEnery (2006).

3 System description

Our system is composed of three major elements, described below:

- Features – common to all models;
- Various models – neural networks or not;
- Ensemble.

3.1 Features

This section describes the features that we used and explains their role. We implemented the following features:

- Number of blacklisted words in the sentence;
- Number of special characters, uppercase characters, etc.;
- A language model taught to recognize offensive and not offensive words.

Blacklisted We used two kinds of blacklisted expressions: "offensive" and "offensive in context". The "offensive in context" expressions are offensive in specific contexts and unoffensive otherwise, e.g. *bloody* or *pearl necklace*. This dictionary was compiled by crowdsourcing and contains about 2,300 words (+ variations). The blacklist consists of swear words, invectives, profanities, slurs and other impolite expressions.

Special characters, uppercase, etc. We checked the graphemic characteristics of the written text and we gave this as a feature to the model. We mainly used the non user related features defined in Founta et al. (2018).

Language model Inspired by the work of Yu et al. (2018), we decided to train a language model on both offensive and non-offensive words. For this purpose, we trained two character based language models, one on the offensive dictionary (described above) and the other from a corpus of non-offensive words. After training them we used the difference in perplexity of each input word as a feature for the model.

3.2 Models

We trained various models and then combined them in an ensemble. This section outlines the models that were part of the ensemble.

Embeddings Both the Neural networks and the machine learning models used embeddings. We used the following embeddings: ELMo (Peters et al., 2018), fastText (Bojanowski et al., 2017), custom embeddings, and Universal Sentence Encoder (Cer et al., 2018). For fastText, we used the 1 million word (300d) vectors trained on Wikipedia 2017, below called fastText 1M.

The custom embedding was built by training a fastText embedding on our corpus. We then combined the 1M fastText embeddings with these custom embeddings using Truncated SVD after concatenating their columns (this was done inspired by the work (Speer et al., 2017)). Building custom embeddings was important for the offensive word classification because the original version of the fastText 1M embeddings contained around 50% of the words in the corpus while after adding the custom embeddings, only 30% of the words were out of the vocabulary. Below, this combination of embeddings is called "combined".

Neural networks We used two types of neural network models:

- LSTM models (Hochreiter and Schmidhuber, 1997);
- Transformer models (Vaswani et al., 2017).

For both models, we used multi-head attention and we tried different embeddings. In most cases, the Transformer models had better results than the LSTM models, and this is what we used in the submissions. The parameters of the models are described in Appendix A.1. In both models, the Features described in Section 3.1 are concatenated to the output of the model.

OpenAI GPT One of the models that we used was the OpenAI GPT (Radford et al., 2018). We used the GPT model in its original form, without changing any parameters. Our results show that this model works very well when there is enough data for finetuning. However, small classes – as in Task 6 Subtask C – pose a problem (see Section 5).

Machine learning models We used two machine learning models:

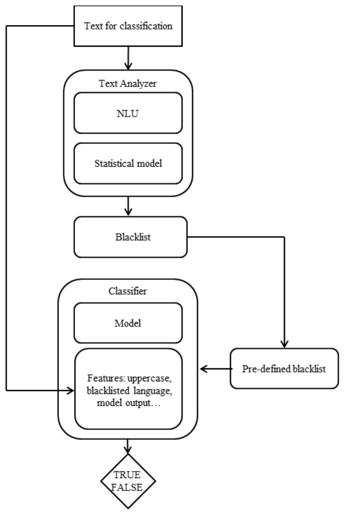


Figure 1: Pipeline for the offensive sentence classifier.

- Random forest;
- SVM.

For these models, we built a pipeline where:

- In a first step we either compute the embeddings of the sentences or get the Td-Idf score.
- In a second step we concatenate the result of the first step with the Features described in Section 3.1 (if used).
- We run the classifier.

As embeddings we used only the Universal Encoder, and with good results.

Ensemble For the ensemble we used a voting classifier with soft voting (based on the probability returned by each model). For each subtask, we show which combination of models gave the best results.

The pipeline for the entire offensive sentence classifier is shown in Figure 1.

4 Data/Datasets

4.1 Preprocessing

Preprocessing plays a crucial role in the analysis of potentially offensive sentences, because most inputs use highly non-standard language. Hence, preprocessing was mainly focused on normalizing the language for simplifying the model work. We applied the following preprocessing:

- Substituting user names with <USER> tokens;
- Removing all links;
- Normalizing words and letters;

- Normalizing spacing and non-standard characters;
- Over/Downsampling of the classes;

After the preprocessing, we split by space and used each token as an input to the models.

Normalizing words and letters We have a dictionary containing common spelling variants of words found in our corpus. We used this to change words to the "canonical" form. Examples of such variants can be seen in Table 2.

Word	Common variants
fuck	fvck, fok, fucc, phuk
nigger	n1gga, n1gr, niigr, nuggah, nigg3r
boob	booob, boooooob
motherfucker	Mutha Fukker, Motha Fuker
ass	a55, 455 ("leetspeak" variants)
assclown	ẫšs̡ĺσẇη (vulgarity obfuscation)

Table 2: Common spelling variants.

Over/Downsampling For each Task/Subtask, we systematically oversampled the classes to obtain a balanced dataset. This was especially important for Task 6/Subtask C, which introduced 3 highly unbalanced classes. For most subtasks we did two things at the same time:

- Downsampled the majority class when there was too much difference from the other classes;
- Oversampled the minority class after downsampling.

4.2 Datasets

In this section we give a high level overview of the datasets we used for training our models for the SemEval-2019 tasks. Detailed statistics are presented in Appendix A.2. For training the model, we used several openly available datasets:

- *Hate Sonar* gathered from Twitter (Davidson et al., 2017);
- 2 related hate speech datasets from Twitter (Waseem and Hovy, 2016), (Waseem, 2016);
- Insulting internet comments (Impermium, 2012);
- Attacking, aggressive, toxic and neutral comments from Wikipedia Talk Pages (Wulczyn et al., 2017);
- *Vulgar Twitter* (Cachola et al., 2018);

our own custom-built corpora and datasets provided by the SemEval organizers.

From the sources listed above, we added a total of 20,399 sentences to the SemEval-2019 corpus for Task 5, and 97,759 sentences to the one for Task 6.

Custom Offensive language corpus Our custom dataset was built by crowdsourcing and by crawling content from the Internet. The dataset is balanced, with 49,179 not offensive and 48,580 offensive comments. Around half of the dataset was labeled by linguists, who were asked to look for "general offensiveness". This could take various forms:

- Expletives, swear-words, offensive terms;
- Rude meaning;
- Meaning that is harsh politically/ethically/ emotionally, and hence expression of hate/ disgust/disrespect;
- Uncomfortable topics related to the human genitals in a gross way;
- Hate speech, sarcasm, sexism, racism, violence, etc.;
- Discussion of drug use or other illegal actions;
- For any other reasons, children should not have access to the sentence.

To each sentence, the linguists assigned one of the three labels:

- OFF – offensive sentence,
- NOT – not offensive sentence,
- Nonsense – random collection of words or non-English (removed from the corpus).

In cases of disagreement between linguists, we chose the most popular label, if applicable, or obtained an expert annotation. We calculated Fleiss' kappa for inter-annotator agreement (Fleiss, 1971), which extends Cohen's kappa to more than two raters (Cohen, 1960). For random ratings Fleiss' $\kappa = 0$, while for perfect agreement $\kappa = 1$. Our κ was equal to 0.62, which falls in the "substantial agreement" category, according to Landis and Koch (1977).

The remaining part of the corpus was assessed automatically with a blacklist-based filter.

Dataset for Task 6 The OLID dataset (Zampieri et al., 2019a) contains Offensive and Not Offensive sentences. The Offensive sentences are further categorized into:

- TIN – targeted insults and threats,
- UNT – untargeted.

and the targeted (TIN) category was further subdivided into:

- IND – individual target,
- GRP – group target,
- OTH – a target that is neither an individual nor a group.

Our full offensive language corpus, described in the previous subsection, was used for this task. The OFF sentences were further annotated for the two categories while the NOT sentences were not further annotated. All the additional classes were added automatically by a wordlist-based annotator.

Dataset for Task 5 The dataset for Task 5 (Basile et al., 2019) contained the classes:

- HATE – hate speech against women or immigrants,
- NOHATE – no hate speech against women or immigrants.

together with other subclasses. Given that we participated in the Task 5 Subtask A, we annotated our corpus only with these two labels. Using a mixture of automated and manual annotation, we were able to add around 30k sentences from our dataset for this task.

5 Results

SemEval In Table 3 we show the average F1 of our models for all the SemEval-2019 Tasks and Subtasks. These results were obtained by using an ensemble of models and in Table 4 we show which model was used inside which ensemble. The acronyms used in the table correspond to:

- **GPT** : OpenAI's GPT model
- **RF**: Random Forest
- **T**: Transformer model
- **U**: Universal encoder
- **EL**: ELMo embeddings
- **CO**: Combined embeddings (see 3.1 for an explanation of this)
- **F**: Features.

Given the short amount of time, during the SemEval competition we were unable to test all the combinations of models and data preparation types to choose the best combination for the Ensemble. We thus selected the models in the ensemble by experimenting with part of the models. This is the main reason why only one model used in Task 5 contains additional features (the TELUF model).

Ensemble	Competition	Macro F1
6-A	Task 6-A	0.80
6-B	Task 6-B	0.69
6-C	Task 6-C	0.63
5-A	Task 5-A	0.51

Table 3: SemEval-2019 results breakdown.

Task	6-A	6-B	6-C	5-A
GPT	✓		✓*	✓
RF	✓	✓	✓	✓
RF+U	✓	✓		✓
T+EL	✓		✓	✓
T+CO+U	✓		✓*	✓
T+EL+U+F				✓

Table 4: Ensemble detail. The models marked with * have been trained with an unbalanced dataset.

After the competition, we tried the models contained in the Ensembles on all the tasks; detailed results are presented in Table 9 in the Appendix. It is important to note though that the results in the Appendix cannot be directly compared with the ones of the SemEval competition because although the models were the same, the Test data was different (the golden data has not been released yet).

From the results we clearly see that we have two "data regimes": in the *low data regime* (Task 6 Subtask B and C), Random forest (with or without the Universal embeddings) is the best choice. However, in *the big(ger) data regime*, Fine tune is the best model. Also in the low data regime each model works best with a different data preparation strategy: GPT with unbalanced data, the Transformer with oversampled and downsampled data while Random forest with oversampled data.

Ablation studies In this part we show the results of ablation studies on the transformer and random forest models. In this study, we want to understand how far the final result was influenced by the linguistically based features and preprocessing we defined in this article. All the results obtained in this section have been computed on a Test set

Model	Task 6 A	Task 6 C
T + CO	0.73	0.44
T + CO + U	0.71	0.52
T + CO + F	**0.75**	0.45
T + CO + U + F	0.74	0.47
RF	0.7	**0.54**
RF + F	0.68	0.43
RF + U	0.72	0.48
RF + U + F	0.69	0.38

Table 5: Macro F1 for selected Transformer models with different combinations of features

created from the Train set shared in the SemEval-2019 tasks (as in Appendix A.3). As we discussed in the previous section, the Tasks were characterized by a "low" (Task 6 C) and a "big" data regime (Task 6 A), thus we compare the ablation study results for these two extreme regimes.

In a first study we wanted to understand how the features influenced the results. For this reason, we tried some combinations of Features, Embeddings and Models on both Task 6 Subtask A and Task 6 Subtask C; the relevant macro F1 results are shown in Table 5. The table shows that, in the big data regime, the Random Forest works best when only the Universal Encoder is used, while the Transformer model improves its performance when the features are added. On the other hand, in the low data regime, we see that the plain Random Forest outperforms all the other combinations. This is probably because the more things we add, the more the model needs to learn, and with little data this is simply not possible.

In a second study we wanted to understand how much the normalization defined in Section 4.1 affected the performance of the model. For this reason, we trained again the best models in Table 5 for both Subtasks with an unnormalized version of the dataset. The results are that for Subtask A, the model **T + CO + F** F1 decreased from 0.75 to 0.73 while for Subtask C, the **RF** F1 decreased from 0.54 to 0.44.

The results of this section seem to point to the fact that the features we added and the normalization we used are beneficial for the performance of the models. Further work will be devoted to understanding this point though.

6 Conclusions

The article presented our approach to making a classifier recognizing offensive expressions in text. It showed how our architecture is suitable for multiple (related) offensive sentence classification tasks. It also showed how we built the features and the data that the model used for learning. Thanks to our system, we were 2^{nd} in the SemEval-2019 Task 6/Subtask C. In the article we also showed with ablation studies that the linguistic features proposed and the embeddings added improve the performance of the models we used.

In the future, we will extend our system to recognize a wider set of features. We are currently working on analyzing the linguistic differences between the offensive corpus and the non-offensive corpus. Specifically, we think that by analyzing the differences, we should be able to build a "white-list" of terms that can be used as features that will help the classifier understand which sentences are less likely to be offensive.

Acknowledgments

We would like to thank G. Knor, P. Przybysz, P. Andruszkiewicz, P. Bujnowski and most of the AI Team in Samsung R&D Institute Poland for all the helpful discussions that made this article possible.

References

Betty van Aken, Julian Risch, Ralf Krestel, and Alexander Löser. 2018. Challenges for toxic comment classification: An in-depth error analysis. arXiv:1809.07572.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics, location = Minneapolis, Minnesota.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. 2016. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings. In *Advances in Neural Information Processing Systems*, pages 4349–4357.

Penelope Brown and Stephen C. Levinson. 1987. *Politeness: Some Universals in Language Usage*.

Isabel Cachola, Eric Holgate, Daniel Preoiuc-Pietro, and Junyi Jessy Li. 2018. Expressively vulgar: The socio-dynamics of vulgarity and its effects on sentiment analysis in social media. In *Proceedings of the 27^{th} International Conference on Computational Linguistics*, pages 2927–2938.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder. arXiv:1803.11175.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Jonathan Culpeper. 2011. Politeness and impoliteness. In *Pragmatics of Society*, pages 391–436.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the Eleventh International AAAI Conference on Web and Social Media (ICWSM 2017)*, pages 512–515.

Jean-Marc Dewaele. 2015. British *bollocks* versus American *jerk*: Do native British English speakers swear more – or differently – compared to American English speakers? *Applied Linguistics Review*, 6(3):309–339.

Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Measuring and mitigating unintended bias in text classification. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*.

Maeve Duggan. 2017. Online harassment 2017.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.

Antigoni-Maria Founta, Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Athina Vakali, and Ilias Leontiadis. 2018. A unified deep learning architecture for abuse detection. arXiv:1802.00385.

Tommi Gröndahl, Luca Pajola, Mika Juuti, Mauro Conti, and N. Asokan. 2018. All you need is "love": Evading hate speech detection. arXiv:1808.09115.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780.

Impermium. 2012. Detecting insults in social commentary.

Timothy Jay and Kristin Janschewitz. 2008. The pragmatics of swearing. *Journal of Politeness Research*, 4:267–288.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying*, pages 1–11.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

Siyuan Li. 2018. Application of recurrent neural networks in toxic comment classification. Master's thesis, University of California, Los Angeles.

Tony McEnery. 2006. *Swearing in English. Bad language, purity and power from 1586 to the present*. Routledge, London and New York.

Ji Ho Park and Pascale Fung. 2017. One-step and two-step classification for abusive language detection on Twitter. arXiv:1706.01206.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT 2018*, pages 2227–2237.

Georgios K. Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Detecting offensive language in tweets using deep learning. arXiv:1801.04433.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical report, OpenAI.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 4444–4451.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Zeerak Waseem. 2016. Are you a racist or am I seeing things? Annotator influence on hate speech detection on Twitter. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 138–142.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the* 26^{th} *International Conference on World Wide Web*, pages 1391–1399.

Xiaodong Yu, Stephen Mayhew, Mark Sammons, and Dan Roth. 2018. On the strength of character language models for multilingual named entity recognition. In *Proceedings of the 2018 conference on empirical natural language processing (EMNLP 2018)*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, and Kai-Wei Chang. 2018. Learning gender-neutral word embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4847–4853.

A Supplementary Materials

A.1 Model parameters

The following parameters were used for all LSTM and Transformer models in the results Section 5:

- keep probability: 0.8;
- LSTM units: 100;
- L2 regularization: 0;
- fully connected size: 256 or 128;
- multihead attention:
  - attention size: 5 or 2,
  - attention head: 4

For GPT, we used a learning rate of 6.25e-5 and an L2 regularization of 0.01.

A.2 Data

In Table 6 we show the amount of data that was contained in our corpus (overall). In Table 7 and 8 we show the data for Task 5 and Task 6. For a description of how these corpora were built and annotated, see Section 4.2.

Source	NOT	OFF	Total
Custom corpus	16,545	12,938	29,483
Kaggle	2,629	3,463	6,092
Twitter	917	23,438	24,355
Wikipedia	29,088	8,741	37,829
Total	49,179	48,580	97,759

Table 6: Statistics for our offensive language corpus. The Kaggle dataset was collected by Impermium (2012). The Twitter dataset was compiled from 4 sources: Davidson et al. (2017), Cachola et al. (2018), Waseem and Hovy (2016) and Waseem (2016). The Wikipedia dataset was collected by Wulczyn et al. (2017).

Class	Total
HATE	16,508
NOHATE	11,154

Table 7: Statistics for the additional corpus for SemEval-2019 Task 5.

A.3 Model results

In this section we show the detailed results of all the models for all the SemEval-2019 tasks. For each Task, we extracted a test set from the Train data released by SemEval. We compared the models to one of the current state of the art defined in

Class	Targeting	Target	Total
OFF	TIN	IND	18,506
		GRP	6,761
		OTH	1,025
		Total	34,669
	UNT	–	6,234
	Total	–	59,837
NOT	–	–	64,773

Table 8: Statistics for the additional corpus for SemEval-2019 Task 6.

Park and Fung (2017); the results shown here are obtained by averaging the best F1 for each class (not a single model). The data by Waseem and Hovy (2016) for comparing to the state-of-the-art model has been kindly shared by the authors of Park and Fung (2017). In the table we marked with

- No additional mark: the normalized data with oversampling and downsampling as described in Section 4.
- **FULL**: the normalized data with oversampling but no downsampling.
- **UNB**: the normalized data without oversampling or downsampling.

The model acronyms are the same as the ones used in Section 5.

Model	5-A	6-A	6-B	6-B FULL	6-C	6-C FULL	6-C UNB	SOTA
RF	0.7	0.62	**0.61**	0.58	0.44	**0.54**	0.45	0.78
RF + F	0.68	0.68	0.59	0.54	0.32	0.43	0.41	-
RF + U	0.72	0.69	0.6	0.55	0.39	0.48	0.46	0.74
GPT	**0.77**	**0.77**	0.58	**0.6**	0.42	0.49	**0.51**	**0.81**
T + CO + U	0.74	0.71	0.58	**0.6**	**0.52**	0.45	0.5	0.73
T + EL	0.73	0.73	0.58	0.58	0.49	0.5	0.45	0.74
SOTA	-	-	-	-	-	-	-	0.78

Table 9: Macro F1 for all the models on all the Tasks and on the state-of-the-art (SOTA) data.

nlpUP at SemEval-2019 Task 6: A Deep Neural Language Model for Offensive Language Detection

Jelena Mitrović, Bastian Birkeneder, Michael Granitzer
Faculty of Computer Science and Mathematics
University of Passau, Germany

jelena.mitrovic@uni-passau.de | birkeneder@fim.uni-passau.de
michael.granitzer@uni-passau.de

Abstract

This paper presents our submission for the SemEval shared task 6, sub-task A on the identification of offensive language. Our proposed model, C-BiGRU, combines a Convolutional Neural Network (CNN) with a bidirectional Recurrent Neural Network (RNN). We utilize word2vec to capture the semantic similarities between words. This composition allows us to extract long term dependencies in tweets and distinguish between offensive and non-offensive tweets. In addition, we evaluate our approach on a different dataset and show that our model is capable of detecting online aggressiveness in both English and German tweets. Our model achieved a macro F1-score of 79.40% on the SemEval dataset.

1 Introduction

The ever-increasing amount of user-generated data introduces new challenges in terms of automatic content moderation, especially regarding hate speech and offensive language detection. User content mostly consists of microposts, where the context of a post can be missing or inferred only from current events. The challenge of automatic identification and detection of online aggressiveness has therefore gained increasing popularity in the scientific community over the last years.

Several recent workshops and conferences such as TRAC (Kumar et al., 2018), ALW2 (Fišer et al., 2018), and GermEval (Wiegand et al., 2018) show the growing importance of this subject. The SemEval 2019 shared task 6 (Zampieri et al., 2019b) further addresses this topic by introducing the Offensive Language Identification Dataset (OLID), which consists of tweets, labeled with a three-level annotation model (Zampieri et al., 2019a). Sub-task A is composed of a binary classification problem of whether a tweet in the dataset is offensive or not. Sub-task B focuses on different categories of offensive language and the goal of sub-task C is to identify the targeted individual of an offensive tweet.

In the following paper, we present our contribution to sub-task A. After the related work section, we outline our conducted experiments in section 3 and further describe the used baseline model, as well as the submitted model. In section 4 we report the results of our experiments on the OLID dataset and the additionally used GermEval dataset. Section 5 discusses our results and section 6 concludes our work and describes possible future work.

2 Related Work

Several methods and models have been presented in literature over the last decade to address the predicament of identifying hate speech, offensive language, and online aggressiveness. In the following section, we present the most notable contributions related to our work.

The tweets collected by Davidson et al. (2017) were divided into Hate, Offensive, and Neither. Their proposed algorithm uses unigram, bigram, and trigram tokens as features, weighted by the respective TF-IDF, as well as Part-of-Speech (POS) tagging and different metrics to determine the readability and sentiment of a tweet. Logistic-regression and linear SVM result in the best performance for a wide range of assessed classifiers. Nobata et al. (2016) collected comments from Yahoo! Finance and News articles over a time period of one year and labeled them as either 'Abusive' or 'Clean'. They experimented with various different features, including n-gram, linguistic, syntactic, and distributional semantics features.

Various approaches utilized deep learning models for text categorization. Zhang et al. (2015) proposed a character-level convolutional network for text classification on large-scale datasets. Their network uses 1-dimensional convolutional filters to extract features from different character embeddings. Gambäck and Sikdar (2017) further experimented with convolutional networks in the context of online hate speech classification. Their research work compares different types of convolutional models, namely character-level, word vectors with a pretrained word2vec (w2v) model, randomly generated word vectors, and w2v in combination with character n-grams. The results of their experiments suggest that w2v embeddings are the most suitable for this task. Zhang et al. (2018) suggest an architecture similar to our network, where a convolutional filter extracts features from pretrained word embeddings. After max pooling, the feature maps are processed using a unidirectional GRU. Their model is compared to a bag-of-n-gram model on various multi-class hate speech datasets and shows promising results. A detailed survey on different architectures, methods and features for offensive language detection is provided by Schmidt and Wiegand (2017).

3 System Description

In addition to Twitter data provided by the organizers of the SemEval shared task, we further evaluate our approach on German tweets from the GermEval (2018) shared task. The OLID dataset contains 13,240 tweets, with 4,400 offensive and 8,840 non-offensive tweets (66.77% offensive, 33.23% non-offensive). Similarly, the GermEval dataset contains 5,009 tweets, divided into 1,688 offensive and 3,321 non-offensive tweets (66.30% offensive, 33.70% non-offensive). To compensate for the imbalanced class distributions and weigh each class equally, we choose the macro averaged F1-score of both classes as our main evaluation metric. From both data sets we use 10% of our tweets as test set. The remaining tweets are split into 90% training set and 10% validation set. We conduct a stratified 10-fold cross-validation on the training and validation set to prevent overfitting and to validate our model.

The pretrained w2v model, which is used to initialize the weights of our embedding layer, resulted from the work of Godin et al. (2015). The w2v model for the GermEval dataset originates from our previous work (2018).

For comparison to our proposed model, a token bag-of-n-gram model composed of unigrams, bigrams, and trigrams weighted by their TF-IDF is used as baseline approach. We subsequently analyze the performance of different classifiers on the resulting feature space.
We have used the packages *keras*, *scikit-learn*, *gensim*, and *nltk* for preprocessing and the implementation of our models.

3.1 Preprocessing

Tweets are first tokenized and converted to lowercase. We constrain repeated character sequences to length 3 and replace all longer character sequences. HTML character encodings are replaced by their corresponding literal or token representation (e.g. '*&amp;*' translates to '*and*'). Tokens are further split if they enclose a set of special characters ('\', '/', '&', '-'). Since hashtags are often used to replace contextually important words mid-sentence, we split hashtags in the actual hash-symbol and the following string to keep the semantic information of a hashtag (e.g. '*Brainless #Liberal Stooge Ocasio-Cortez*').

3.2 Baseline Model

A TF-IDF bag-of-words model as baseline approach is chosen to evaluate the performance of our model. We limit our feature space to the 10,000 most frequently used unigrams, bigrams, and trigrams in a corpus. Furthermore, we stem each token in the preprocessing phase and remove stopwords. We compare the performance of several classifiers, namely multinomial Naive Bayes (NB), SVM, Decision Tree (DT), and Logistic Regression (LogR) and conduct a grid search to optimize our hyper-parameters.

3.3 C-BiGRU

After the preprocessing step, we construct a dictionary which maps all unique tokens to their number of occurrences in the respective corpus. Tokens which appear only once in a corpus are disregarded and treated as unknown token. As a next step, we construct the weighting matrix $W^{m \times dim}$ for our embedding layer, where dim is the dimension of the used w2v model and m the number of unique tokens $t_i, i \in \{1, ..., m\}$. The word vector of t_i is stored in W if the token is represented in the w2v model. If t_i has no pretrained word vector, we generate a random vector drawn from the uniform distribution within $\left[-\sqrt{\frac{6}{dim}}, \sqrt{\frac{6}{dim}}\right]$ as suggested by He et al. (2015). We fix the maximum length of a sentence to 150 tokens, longer sequences are clipped at the end and shorter sequences are padded with a masking token.

The convolutional layer of our classifier consists of $(k \times 128)$ 1-dimensional filters, where k is the number of different window sizes. These window sizes range from 2 to 5 and allow the extraction of n-gram features. The padding of the input is kept constant, resulting in the same output sequence length as the input. We further choose ReLu as activation function. The resulting feature maps are concatenated and passed towards the recurrent layer.

Gated Recurrent Units (GRU) as initially proposed by Cho et al. (2014) are used in RNNs to capture long-term dependencies of input sequences. Similar to Long Short-Term Memory (LSTM) units (Hochreiter and Schmidhuber, 1997) GRU are able to overcome the vanishing gradient problem by using a gating mechanism. GRU have shown to achieve comparable results to LSTM in sequence modeling tasks and are able to outperform the latter on smaller data sets (Chung et al., 2014). The recurrent layer in our model consists of a bidirectional GRU, where the concatenated feature maps, which resulted from the convolutional layer, are used as input for the GRU layer. Simultaneously, the reversed copy of the input sequence is used for the second GRU layer. Both GRU layers return a hidden state for each processed feature map. The output of both layers is then concatenated. We set the length of the returned hidden states to 64 for both layers, resulting in an output space of (150×128) neurons.

Afterwards, a global max pooling layer reduces the output space to (1×128) nodes. The following fully-connected layer consists of 32 neurons, which connect to a single output neuron. The output neuron utilizes the *sigmoid* activation function.

To additionally prevent overfitting, we include two dropout layers with a dropout rate of 0.2; one after the embedding layer and another one after the fully-connected layer. Furthermore, we adopt early stopping and use 10% of the training data as validation split. We use cross entropy as error function for our model and the optimizer 'adam' to update our network weights (Kingma and Ba, 2014). The batch size for the gradient update is set to 32. A schema of our proposed model is illustrated in Figure 1.

4 Results

For the comparison model, the SVM performs best on the OLID dataset with an F1-score of 70.22% averaged over a 10-fold cross-validation. The SVM also shows the best results on the GermEval dataset with an F1-score of 66.61%. The evaluation on the test set results in 66.78% F1-score for the GermEval gold test set. The evaluation of the baseline model for the OLID gold test set is not possible at the time of writing, since the gold test data have not yet been released.

The C-BiGRU achieved a 76.28% F1-score on the OLID and a 71.13% F1-score on the GermEval dataset on average over a 10-fold cross-validation. On the OLID gold test set, our model achieved an F1-score of 79.40%. The evaluation on the GermEval gold test data resulted in a 72.41% F1-score. An overview of all results can be found in Table 1. Figure 2 shows the confusion matrix of our submitted predictions for the SemEval shared task.

	Baseline		C-BiGRU	
	CV	gold	CV	gold
OLID	70.22%	-	76.28%	79.40%
GermEval	66.61%	66.78%	71.13%	72.41%

Table 1: All results in table form (CV = cross-validation; gold = gold test set).

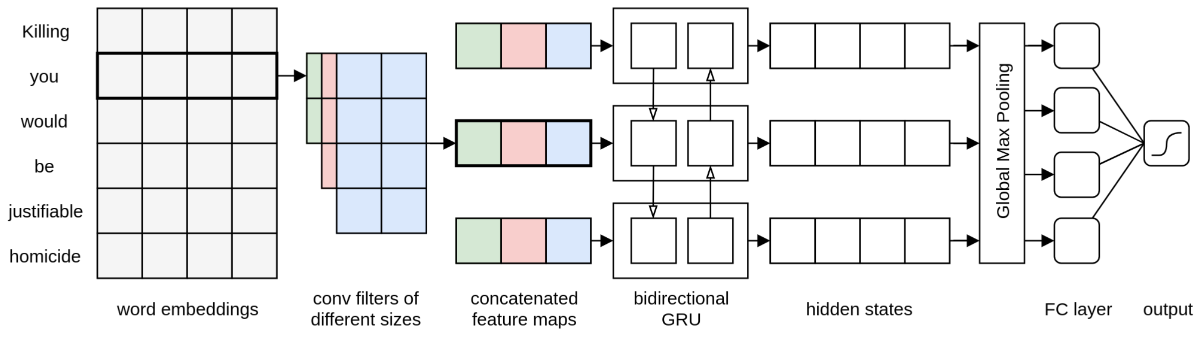


Figure 1: Representation of the proposed classifier.

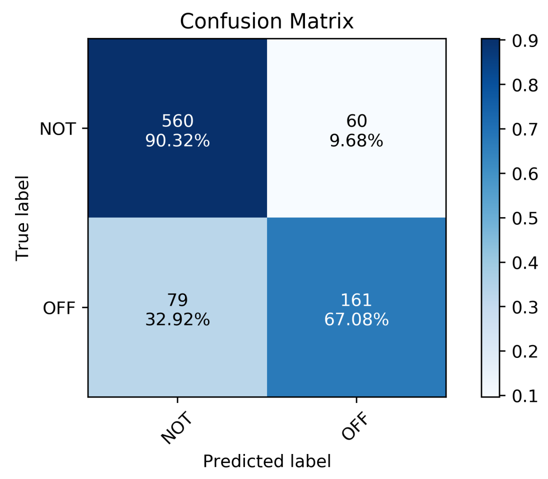


Figure 2: Confusion Matrix of the OLID gold test set, sub-task A. Depicted are instances and normalized values.

5 Discussion

The presented model continues our work on the identification of offensive German tweets (2018). We were able to improve our proposed model by adjusting the architecture of the recurrent layer in our neural network. By using a bidirectional GRU instead of a unidirectional LSTM, we are able to capture past and future information about the input sequence and exploit the better performance of GRU networks on smaller datasets. Furthermore, we return the hidden states for each feature map instead of returning only the last hidden state. This allows us to extract higher-level sequentially dependent features from each concatenated feature map.

Our experiments show that our suggested model outperforms the baseline model on both datasets. The difference between the F1-scores for the English and German dataset might be attributed to the smaller size of the German training set, which contains only about 5,000 tweets. The discrepancy between the results of our cross-validation and achieved score on the OLID test set might be explained by the small amount of test tweets, which may lead to imprecise results for the submitted runs.

By utilizing w2v as features, we are able to limit extensive and language specific preprocessing.

> “@USER Lolol God he is such an a**hole.”

In this example, the vector representation of “a**hole” has a high cosine similarity (0.63) to the vector representation of “asshole”, which allows our model to classify this tweet as offensive. On the contrary, our approach falls short when confronted with indirect insults.

> “@USER @USER Im sure the air that he is breathing is also bad.”

Our model wrongly predicts a non-offensive tweet in this instance.

The detection of offensive, hateful, racist, and/or sexist user behavior in social media still proves to be a challenge. Even for humans, it can be problematic to identify offensive microposts, since these posts can be ambiguous and dependant on the personal mindset of a reader. Ross et al. (2017) show that it can be difficult to measure the agreement of annotators about hate speech in the light of the European refugee crisis. They conclude that instead of a classification problem, a regression model with an average offensiveness score of multiple annotators might be more suitable for this task. Furthermore, it can be difficult to grasp the full context of an arbitrary tweet. With only excerpts of a conversation, the context and true intention of the author may be difficult to determine.

6 Conclusion and Future Work

In this paper, we describe our submitted model for the SemEval shared task 6 and evaluation methods for the identification of online aggressiveness in social media microposts. Our model achieves good results in the two evaluated datasets. For the OLID dataset which contains English tweets, a macro F1-score of 79.40% is reached, while our network resulted in an F1-score of 72.41 % on the GermEval dataset, which consists of German tweets.

We plan to evaluate our approach on more datasets to further investigate the potential of our model for different languages. One such set is the TRAC dataset, which contains aggression-annotated Facebook posts and comments in Hindi. Furthermore, we want to examine whether additional features such as character-level embeddings or POS tagging will improve our results. Inclusion of figurative language detection has proved to enhance many NLP tasks, such as argument mining and so-called hidden hate speech (Mitrović et al., 2017), which is also one of our future directions.

References

Bastian Birkeneder, Jelena Mitrović, Julia Niemeier, Leon Teubert, and Siegfried Handschuh. 2018. upInf - Offensive Language Detection in German Tweets. In *Proceedings of the GermEval 2018 Workshop*, pages 71 – 78.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078v3.*

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555.*

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. *arXiv preprint arXiv:1703.04009.*

Darja Fišer, Ruihong Huang, Vinodkumar Prabhakaran, Rob Voigt, Zeerak Waseem, and Jacqueline Wernimont. 2018. Proceedings of the 2nd workshop on abusive language online (alw2). In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2).*

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using convolutional neural networks to classify hatespeech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Fréderic Godin, Baptist Vandersmissen, Wesley De Neve, and Rik Van de Walle. 2015. Multimedia lab @ acl wnut ner shared task: Named entity recognition for twitter microposts using distributed word representations. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 146–153.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Ritesh Kumar, Atul Kr. Ojha, Marcos Zampieri, and Shervin Malmasi. 2018. Proceedings of the first workshop on trolling, aggression and cyberbullying (trac-2018). In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018).* Association for Computational Linguistics.

Jelena Mitrović, Cliff O'Reilly, Miljana Mladenović, and Siegfried Handschuh. 2017. Ontological representations of rhetorical figures for argument mining. *Argument & Computation*, 8(3):267–287.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153. International World Wide Web Conferences Steering Committee.

Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Wojatzki. 2017. Measuring the reliability of hate speech annotations: The case of the european refugee crisis. *arXiv preprint arXiv:1701.08118.*

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the germeval 2018 shared task on the identification of offensive language. *Austrian Academy of Sciences, Vienna September 21, 2018.*

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL.*

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval).*

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting hate speech on twitter using a convolution-gru based deep neural network. In *European Semantic Web Conference*, pages 745–760. Springer.

Pardeep at SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media Using Deep Learning

Pardeep Singh
School of Computer & Systems Sciences
Jawaharlal Nehru University
New Delhi-110067
pardeepsinghinfo@gmail.com

Satish Chand
School of Computer & Systems Sciences
Jawaharlal Nehru University
New Delhi-110067
schand20@gmail.com

Abstract

The rise of social media has made information exchange faster and easier among the people. However, in recent times, the use of offensive language has seen an upsurge in social media. The main challenge for a service provider is to correctly identify such offensive posts and take necessary action to monitor and control their spread. In this work, we try to address this problem by using sophisticated deep learning techniques like LSTM, Bidirectional LSTM and Bidirectional GRU. Our proposed approach solves 3 different Sub-tasks provided in the SemEval-2019 task 6 which incorporates identification of offensive tweets as well as their categorization. We obtain significantly better results in the leader-board for Sub-task B and decent results for Sub-task A and Sub-task C validating the fact that the proposed models can be used for automating the offensive post-detection task in social media.

1 Introduction

Social media has revolutionized the way of communication among the people. It is an instant communication medium which connects people all over the world and shares their views. But, some people misuse this freedom by using the offensive language through posts or comments to defame, insult or target an individual or a group of individuals. The mainstream media have reported various cases of suicide and depression due to trolling and cyberbullying in social media. Hence it becomes worrisome for the corporates, government organizations and security agencies to either stop or mitigate this type of behavior of the users. Manually it is impossible to check the negative behavior of users due to the volume, velocity and variety of data coming from the social networks. Hence there is an utmost need to develop a system which automatically identifies and categorizes the offensive language in social networks. To tackle these issues SemEval-2019 (Zampieri et al., 2019b) aimed exactly at that need and organized a task in identifying and categorizing offensive language in social media. This task is divided into three Sub-tasks.
Sub-task A - Offensive language identification.
Sub-task B - Categorization of offense types.
Sub-task C - Offense target identification.
All the three Sub-tasks are related to each other. In Sub-task A, we have to identify whether a given set of tweets is offensive or not. It is a binary classification task based on tweet text. In Sub-task B, the main challenge is to categorize the tweets which are offensive in Sub-task A into targeted or untargeted. Sub-task C is comparatively challenging than other two Sub-tasks due to the multi-class nature. Its goal is to identify the tweets which are targeted in Sub-task B and categorized those tweets into individual, group or others.
Our approach for the SemEval-2019 task 6 (identifying and categorizing offensive language in social media) comprises of deep learning models: Bidirectional LSTM, Bidirectional GRU and standard LSTM. These are popularly used deep learning sequence models applied in many text classification tasks. We used the pre-trained word level embedding GloVe (Global Vectors for Word Representation) to get vector representations for words that appeared in tweets and used these representations as features for training the models. To check the performance of models, 10 fold cross-validation was applied on the given training data. We compared the results of the above-mentioned models with various baselines such as Logistic Regression, Support Vector Machine, Gradient Boosting and XGBoost. The baseline models are reasonably good but they have poor classification Accuracy as compared to deep learning models. This paper presents the description of our approaches and results for SemEval-2019 task 6.

2 Related Work

This section discusses some existing work related to identifying and categorizing offensive language in social media. Researchers have applied various computational methods to deal with hate speech, aggression, offensive language, racist and sexist language, and cyberbullying.

Hate Speech: Detection of hate speech is modeled in (Zhang et al., 2018). The authors applied CNN and GRU deep neural networks along with pre-trained Google Word2vec word embedding to detect the hate speech on Twitter. (Zhang and Luo, 2018) proposed Skip Gram Extraction CNN (SKIP-CNN) deep neural network model to identify hate speech present in social media text. It is discussed in this paper that hate speech lacks distinctive and unique features in a dataset which is hard to discover. The proposed model serves as a feature extractor for capturing the semantics of hate speech in social media.

Aggression: A method to detect aggression in social media is proposed in (Madisetty and Desarkar, 2018). The authors applied CNN, LSTM and Bidirectional LSTM on Facebook comment dataset. The output of these three deep learning models are used as an input to the majority based ensemble method for detection of aggression in social media. Another paper (Kumar et al., 2018) presents the system description report of shared task on identification of aggression in social media as a part of the 1st workshop on trolling, aggression and cyberbullying (TRAC1). The aggression annotated dataset of Facebook posts and comments in English and Hindi language were provided to the participants for training and validation. Six models out of the top ten best performing models were trained using LSTM, Bidirectional LSTM, CNN, and RNN deep neural networks.

Racist and Sexist Language: (Davidson et al., 2017) focused on classifying homophobic and racist tweets as hate speech and sexist remarks tweets as offensive. They use Logistic Regression with L2 regularization to predict the class membership. (Pitsilis et al., 2018) proposed an ensemble LSTM deep learning classifier that utilizes the user behavior metric to show each user viewpoint towards racism and sexism captured by their tweeting history.

Cyberbullying: (Dadvar et al., 2013) studied about the Cyberbullying detection. They combine individual comments, user characteristics and user profile information for training the Support Vector Machine classifier. It is also reported that the addition of user history with text features improves cyberbullying detection accuracy. (Rafiq et al., 2018) proposed a multi-stage cyberbullying detection mechanism by two novel components. First is dynamic priority scheduler which drastically reduces the classification time, and second is incremental classification method which is highly responsive regarding time to raise alerts.

Until now there have been many publications and studies on offensive language, aggression and hate speech in social media. Examples include (Wiegand et al., 2018), (ElSherief et al., 2018) and (Fortuna and Nunes, 2018). All these methods have some pros and cons associated with them. Therefore this paper proposed the idea of using deep learning sequence models for better accuracy in results for SemEval-2019 task 6.

3 Methodology and Data

In this section, we first describe the dataset used in the competition and then we explain the description of approaches used for solving the problem.

3.1 Dataset Used

The dataset provided by the task organizers is OLID (Offensive Language Identification). The details of data and annotation are available in (Zampieri et al., 2019a). For Sub-task A, this dataset contains tweets labeled into the following two categories: offensive (OFF) and not offensive (NOT). For Sub-task B, tweets are labeled into the following two categories: targeted input (TIN) and untargeted (UNT). For Sub-task C, the given tweets are classified into the following three categories: group (GP), individual (IND) and others (OTH). Out of 13,240 training samples of Sub-task A, 4404 samples have been allocated to Sub-task B and 3,877 samples have been allocated to Sub-task C. All the tweets are in English language. The statistics of the dataset and some instances of tweets with their labels are shown in Table 1 and Table 2.

	Training Set samples	Testing Set samples
Sub-task A	13240	860
Sub-task B	4404	240
Sub-task C	3877	213

Table 1: Statistics of the offensive dataset

Tweet	Sub-task A	Sub-task B	Sub-task C
Its not my fault you support gun control.	NOT	-	-
Someone should'veTaken" this piece of shit to a volcano.	OFF	UNT	-
you are a lying corrupt traitor!!! Nobody wants to hear anymore of your lies!!! #DeepStateCorruption.	OFF	TIN	IND
Kind of like when conservatives wanna associate everyone to their left as communist antifa members?	OFF	TIN	GRP
why report this garbage. We don't give a crap.	OFF	TIN	OTH

Table 2: Some tweets from the training dataset with their labels.

3.2 Methodology

Here we discuss our proposed approach in details. Our initial approach was to check with standard machine learning algorithms like Logistic Regression (Hosmer Jr et al., 2013), Random Forest (Xu et al., 2012), Support Vector Machines (Chang and Lin, 2011), XGBoost (Chen and Guestrin, 2016) and Gradient Boosting (Natekin and Knoll, 2013). We use TF-IDF vectorization for vectorizing our text and then apply the above-mentioned algorithms for the model development. Performance of these algorithms were not quite acceptable as it gave low Accuracy in results. To overcome above mentioned issues, we use deep learning algorithms for classifying the text. First we convert the text into vector representations with the help of GloVe (Pennington et al., 2014) word level embeddings and then use these representations as an input to the deep learning models described in the subsequent sections for classification tasks.

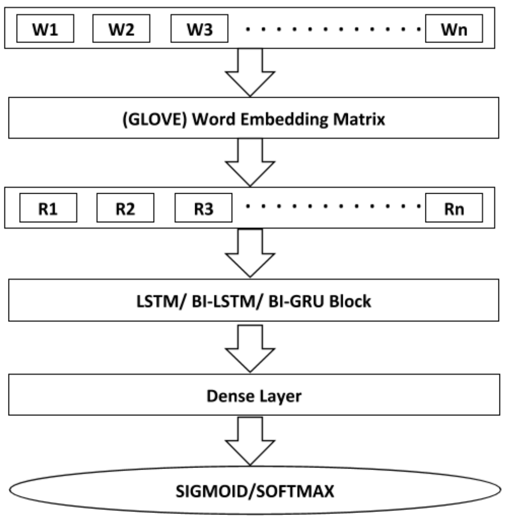


Figure 1: Architecture of the proposed deep learning models.

The multi-layered architecture of our approach presented in Figure 1. It comprised of various components in the form of layers. Since the data is in the form of text and the first step is to vectorize the text. To achieve this, we first make the tokens of the text W1, W2, W3,..., Wn and apply pre-trained GloVe word embeddings to get vector representations R1, R2, R3,..., Rn from it. Next layer can be LSTM, Bidirectional LSTM or Bidirectional GRU Block described in the next subsections. To overcome the problem of overfitting, we add a small amount of dropout. Finally, we use a Dense layer and Softmax/Sigmoid layer to get the output of the models.

3.2.1 Long Short Term Memory (LSTM)

We first try LSTM (Hochreiter and Schmidhuber, 1997) which have been used successfully in many text classification tasks (Madisetty and Desarkar, 2018). LSTM is special kind of RNN which captures the long contexts and long-range dependencies very efficiently in the sentences and takes care of the vanishing gradient problem of RNN (Lipton et al., 2015) with the help of carefully regulated structures named gates. The main components of the LSTM model are input gate, forget gate, output gate and candidate memory state. All these gates are single layered neural networks with the Sigmoid activation function except candidate memory state which uses tanh as the activation function.

3.2.2 Gated Recurrent Unit (GRU)

Gated Recurrent Unit (Tjandra et al., 2016) is an improvisation over LSTM. They also take care of the vanishing gradient problem of the RNN and tries to capture long-range connections better but with a less number of gates than LSTM. This leads to a less amount of parameters for the model which enables a faster and efficient model development in comparison to the LSTM based model.

The main components of GRU are reset gate, update gate and current memory content. Similar to LSTM, both reset gate and update gate are single layered neural networks with the Sigmoid activation function except current memory content which use tanh as the activation function. The basic function of the reset gate is to determine how much of the past information to be lost whereas the update gate decides how much of the information the model should pass to the next states.

3.2.3 Bidirectional LSTM and GRU

Both LSTM and GRU uses sequential information of the textual data for the processing and capture much longer range dependencies. But, the catch is that they use the sequence of only one direction while the Bidirectional version of the same considers a reverse copy of the provided input. In certain problems, this reversal helps to a better feature understanding and improved model performance.
In our work, we mainly use standard LSTM and Bidirectional version of both LSTM and GRU for the model developments. The detailed experimental setup is described in the next section.

4 Experimental Setting

For implementing the models, we use Keras (Ketkar, 2017) and Scikit-learn (Pedregosa et al., 2011) python framework libraries. The experimental details and model configuration are shown in Table 3. For the effectiveness of models, we add a small proportion of dropout. For GRU model, we specify the number of Recurrent Units. In terms of training, we use categorical cross Entropy as a loss function with ADAM as the optimization function. All the models are tested using 10 fold cross-validation.

Model Configuration	**Value**
sentences_length	32
batch_size	64
recurrent_units (for GRU)	64
dense_size	16
dropout_rate	0.5
number of epochs	300

Table 3: Configuration of the proposed models.

4.1 Impact of Batch Size on Model Performance

We checked our proposed approach with three different batch_sizes 64, 128 and 256 to check its impact on model performance. It is found experimentally that batch_size 64 provides optimal results.

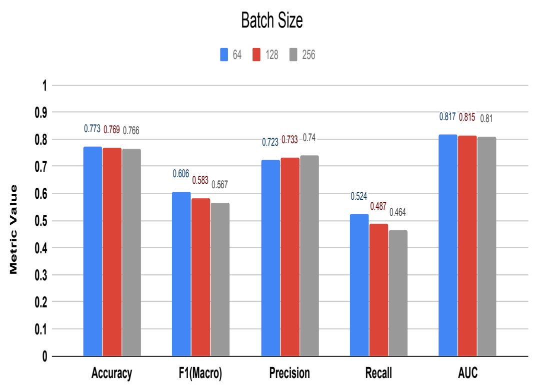


Figure 2: Performance comparisons with different batch Sizes.

Performance metrics: The official evaluation metric for all the three Sub-tasks are the macro-averaged F1 score. For additional analysis, we use the Accuracy, Precision, Recall and ROC-AUC.

5 Results and Discussions

This section contains the detailed experimental results that we performed on the proposed models including the baselines. It is quite familiar that multiple baselines approaches are helpful for comparing the performance of models on validation sets. To observe this, we apply various computational models on the training data released for Sub-task A so that we figure out which models give better results on the training data. Table 4 presents each model results in terms of Accuracy, F1(Macro), Precision, Recall and Roc-Auc score. It is evident from this Table that the deep learning models like LSTM, Bidirectional LSTM and Bidirectional GRU with GloVe word embeddings outperformed TF-IDF based machine learning algorithms. The LSTM model provides better results in terms of Accuracy among all the models. Bidirectional LSTM provides better results in terms of F1 macro and the Random Forest with TF-IDF gives better results in terms of Precision. The Bidirectional GRU provides better results for Recall matrix. The standard LSTM and Bidirectional LSTM performs equally good in terms of ROC-AUC.

Classifier	Accuracy	F1(Macro)	Precision	Recall	ROC-AUC
Logistic Regression + TF-IDF	0.7610	0.5563	0.5070	0.4463	0.6832
Random Forest + TF-IDF	0.7567	0.5189	**0.7718**	0.3908	0.6662
SVM with linear Kernel + TF-IDF	0.7658	0.5579	0.7613	0.4403	0.6853
Xgboost + TF-IDF	0.7323	0.5248	0.6493	0.4403	0.6601
Gradient Boosting + TF-IDF	0.7525	0.5750	0.6785	0.4988	0.6897
BI-LSTM + GloVe	0.7686	**0.6089**	0.7026	0.5459	**0.8100**
BI-GRU + GloVe	0.7524	0.6021	0.6501	**0.5672**	0.7963
LSTM + GloVe	**0.7695**	0.5942	0.7191	0.5081	**0.8100**

Table 4: Results of the proposed deep learning approaches including baselines on the Sub-task A training data using 10-fold cross validation

5.1 Results for Sub-task A

The official results of our proposed models on the test set for Sub-task A is shown in Table 5. As it is evident from these results that Bidirectional GRU performed better than other two deep learning models with F1 Score of 0.69. To analyze the correct label of a tweet, we also show the confusion matrix which shows correct class predictions along diagonal lines. Our team ranked 74 out of 104 participating teams.

System	**F1 (macro)**	**Accuracy**
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
BI-LSTM	0.6617	0.7535
BI-GRU	**0.6992**	**0.7744**
LSTM	0.6785	0.7477

Table 5: Results for Sub-task A (Binary Classification)

5.2 Results for Sub-task B

As comparison to Sub-task A, the official results for Sub-task B shown in Table 6 are significantly better. Our team ranked 7 out of 76 participating teams with F1 Score of 0.69. Again the Bidirectional GRU outperforms both LSTM and Bidirectional LSTM deep learning models in terms of F1 and Accuracy.

System	**F1 (macro)**	**Accuracy**
All TIN baseline	0.4702	0.8875
All UNT baseline	0.1011	0.1125
BI-LSTM	0.6511	0.8958
BI-GRU	**0.6997**	**0.9**
LSTM	0.6455	0.8917

Table 6: Results for Sub-task B (Binary classification)

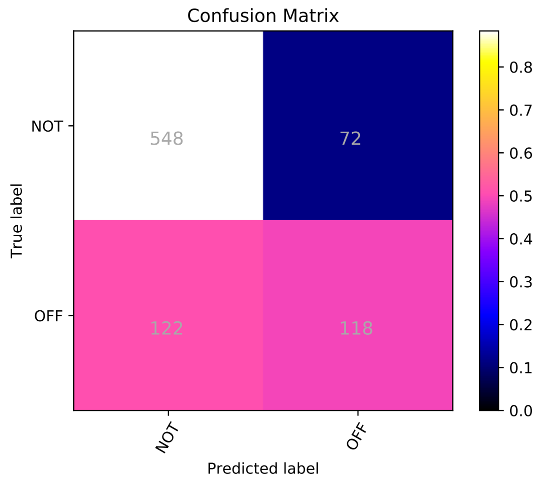


Figure 3: Confusion Matrix shows results for Sub-task A using Bidirectional GRU

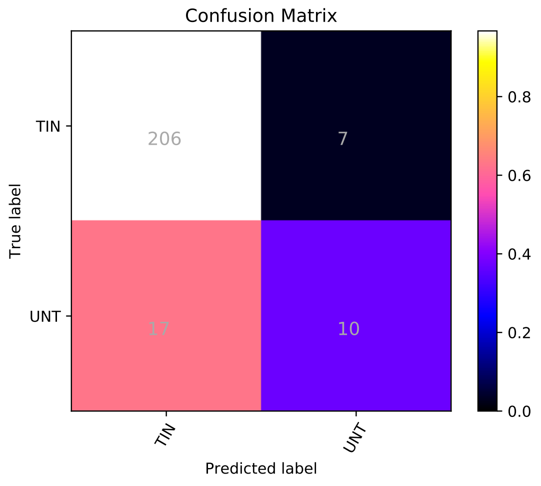


Figure 4: Confusion Matrix shows results for Sub-task B using Bidirectional GRU

5.3 Results for Sub-task C

Table 7 presents our results on the test set for Sub-task C. For this multi-class classification challenge, the results are lower as compared to other two Sub-tasks. All the participating teams had a lower performance with highest F1 score of 0.66, demonstrating the difficulty of the Sub-task. Our team ranked 41 out of 65 participating teams and Bidirectional LSTM give better results with F1 score of 0.49.

System	F1 (macro)	Accuracy
All GRP baseline	0.1787	0.3662
All IND baseline	0.2130	0.4695
All OTH baseline	0.0941	0.1643
BI-LSTM	**0.4903**	**0.6056**
BI-GRU	0.4635	0.5775
LSTM	0.4810	0.6197

Table 7: Results for Sub-task C (Multi-Class Classification)

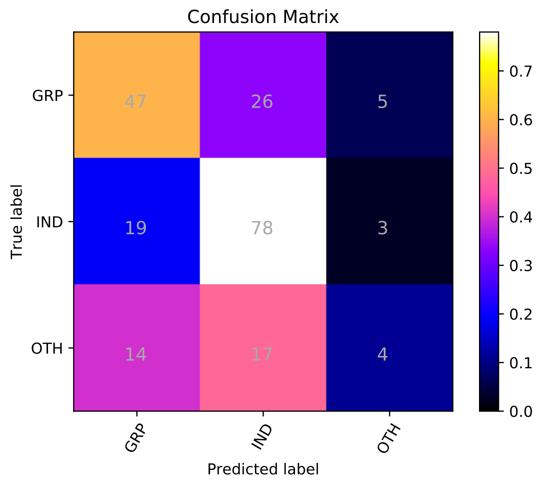


Figure 5: Confusion Matrix shows results for Sub-task C using Bidirectional LSTM

5.4 Class Label results

Besides the combined results of our proposed models on the test set for three Sub-tasks, we also present per class results in Tables 8, 9 and 10. The results in these tables show how well our models performed on each class label.
Table 8 shows each class label results for Sub-task A which comprises of two classes offensive (OFF) and not offensive (NOT). Bidirectional GRU performed better on both classes with F1 score of 0.54 and 0.84 respectively validating the fact that offensive class are relatively difficult to classify.
Table 9 shows each class label results for Sub-task B which comprises of two classes targeted input (TIN) and untargeted (UNT). Bidirectional GRU performed better on both classes with F1 score of 0.94 and 0.45 respectively which shows that untargeted class are much harder to classify.
Table 10 shows each class label results for Sub-task C which is a multi-class classification challenge having three classes individual (IND), group (GRP) and others (OTH). LSTM provides better results with F1 score of 0.72 and 0.63 for both (IND) and (GRP) classes while Bidirectional LSTM performed better on (OTH) class with F1 score of 0.17, justifies that (OTH) class is relatively harder to classify.

6 Conclusion

In this paper, we address the challenge of identification of offensive tweets as well as their categorization. Our proposed approach comprises of three deep learning based techniques for efficient classification of offensive posts in social media. In this work, we show that applying word embedding over social media text followed by the application of a sequence to sequence models like LSTM, Bidirectional LSTM and Bidirectional GRU leads to a better classification of the text. This proposed approach can also be incorporated in an end-to-end framework. Overall, our approach provides an efficient way of text classification in social media. For future work, we want to include character-based embeddings along with pre-trained word level embeddings for better representation of text. Also, the addition of attention layer to the deep networks sometimes increases performance even further.

Acknowledgements

The aforementioned work is funded by the University Grants Commission (UGC), New Delhi under Senior Research Fellowship (SRF). We would also like to acknowledge Dr. Srijith P.K. and Uddipta Bhattacharjee from the Department Of Computer Science IIT Hyderabad for their support and valuable suggestion.

	OFF			NOT		
	P	R	F1	P	R	F1
BI-LSTM	0.5814	0.4167	0.4854	0.7965	**0.8839**	0.8379
BI-GRU	**0.6211**	0.4917	**0.5488**	**0.8179**	**0.8839**	**0.8496**
LSTM	0.5520	**0.5083**	0.5293	0.8153	0.8403	0.8276

Table 8: Shows per-class performance of our proposed models for Sub-task A.

	TIN			UNT		
	P	R	F1	P	R	F1
BI-LSTM	0.9123	0.9765	0.9433	0.5833	0.2593	0.3590
BI-GRU	**0.9238**	0.9671	**0.9450**	**0.5882**	**0.3704**	**0.4545**
LSTM	0.9119	**0.9718**	0.9409	0.5385	0.2593	0.3500

Table 9: Shows per-class performance of our proposed models for Sub-task B.

	IND			GRP			OTH		
	P	R	F1	P	R	F1	P	R	F1
BI-LSTM	0.6446	0.7800	0.7059	0.5875	0.6026	0.5949	**0.3333**	**0.1143**	**0.1702**
BI-GRU	0.6389	0.6900	0.6635	0.5667	**0.6538**	0.6071	0.2000	0.0857	0.1200
LSTM	**0.6508**	**0.8200**	**0.7257**	**0.6575**	0.6154	**0.6358**	0.1429	0.0571	0.0816

Table 10: Shows per-class performance of our proposed models for Sub-task C.

References

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM.

Maral Dadvar, Dolf Trieschnigg, Roeland Ordelman, and Franciska de Jong. 2013. Improving cyberbullying detection with user context. In *Advances in Information Retrieval*, pages 693–696. Springer.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Lstm can solve hard long time lag problems. In *Advances in neural information processing systems*, pages 473–479.

David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. 2013. *Applied logistic regression*, volume 398. John Wiley & Sons.

Nikhil Ketkar. 2017. Introduction to keras. In *Deep Learning with Python*, pages 97–111. Springer.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Zachary C Lipton, John Berkowitz, and Charles Elkan. 2015. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*.

Sreekanth Madisetty and Maunendra Sankar Desarkar. 2018. Aggression detection in social media using deep neural networks. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 120–127.

Alexey Natekin and Alois Knoll. 2013. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier

Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Georgios K Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*.

Rahat Ibn Rafiq, Homa Hosseinmardi, Richard Han, Qin Lv, and Shivakant Mishra. 2018. Scalable and timely detection of cyberbullying in online social networks. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pages 1738–1747. ACM.

Andros Tjandra, Sakriani Sakti, Ruli Manurung, Mirna Adriani, and Satoshi Nakamura. 2016. Gated recurrent neural tensor network. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 448–455. IEEE.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval*.

Baoxun Xu, Joshua Zhexue Huang, Graham Williams, Qiang Wang, and Yunming Ye. 2012. Classifying very high-dimensional data with random forests built from small subspaces. *International Journal of Data Warehousing and Mining (IJDWM)*, 8(2):44–63.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ziqi Zhang and Lei Luo. 2018. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *Semantic Web*, (Preprint):1–21.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

SINAI at SemEval-2019 Task 6: Incorporating lexicon knowledge into SVM learning to identify and categorize offensive language in social media

Flor Miriam Plaza-del-Arco, M. Dolores Molina-González, M. Teresa Martín-Valdivia, L. Alfonso Ureña-López

Department of Computer Science, Advanced Studies Center in ICT (CEATIC)
Universidad de Jaén, Campus Las Lagunillas, 23071, Jaén, Spain
{fmplaza, mdmolina, maite, laurena}@ujaen.es

Abstract

Offensive language has an impact across society. The use of social media has aggravated this issue among online users, causing suicides in the worst cases. For this reason, it is important to develop systems capable of identifying and detecting offensive language in text automatically. In this paper, we developed a system to classify offensive tweets as part of our participation in SemEval-2019 Task 6: OffensEval. Our main contribution is the integration of lexical features in the classification using the SVM algorithm.

1 Introduction

In recent years, with the emergence of social media, the user-generated content on the Web has grown exponentially. This content has the potential to be transmitted quickly, reaching anywhere in the world in a matter of seconds. Due to the exchange of ideas between users, we find not only positive comments, but also a wide diffusion of aggressive and potentially harmful content. Consequently, this type of remarks affects millions of online users. In fact, it has been reported that these incidents have not only created mental and psychological agony to the online users, but have forced people to deactivate their accounts and, in severe cases like cyberbullying, to commit suicides (Hinduja and Patchin, 2018). One of the strategies used to deal with aggressive behavior in social media is to monitor or report this type of content. However, this strategy is not entirely feasible due to the huge amount of data that is generated daily by users. Therefore, it is necessary to develop systems capable of identifying this type of content on the Web.

In order to tackle this problem, firstly it is important to define the toxic language. The toxic language can be broadly divided into two categories: hate speech and offensive language (Cheng, 2007; Davidson et al., 2017; Gaydhani et al., 2018). According to Cambridge Dictionary, hate speech is defined as "public speech that expresses hate or encourages violence towards a person or group based on something such as race, religion, sex, or sexual orientation". Offensive language is defined as the text which uses hurtful, derogatory or obscene terms made by one person to another person.

In this paper, we present the system we developed as part of our participation in SemEval-2019 Task 6 OffensEval: Identifying and Categorizing Offensive Language in Social Media) (Zampieri et al., 2019b). In particular, we participated in subtask A: Offensive language identification. It is a binary classification task and consists of identifying if a post contains or not offense or profanity language.

The rest of the paper is structured as follows. In Section 2, we explain the data used in our methods. Section 3 introduces the lexical resources used for this work. Section 4 presents the details of the proposed systems. In Section 5, we discuss the analysis and evaluation results for our system. We conclude in Section 6 with remarks on future work.

2 Data

To run our experiments, we used the English dataset provided by the organizers in SemEval19 Task 6 OffensEval: Identifying and Categorizing Offensive Language in Social Media (Zampieri et al., 2019a).

The datasets contain tweets with five fields. Each tweet comprises an identifier (id), the tweet text (tweet), field for subtask A (subtask_a), field for subtask B (subtask_b) and field for subtask C (subtask_c). Since we have only participated in sub-task A, we are interested in the fields id, tweet and subtask_a.

In sub-task A, we are interested in the identification of offensive tweets and tweets containing any form of (untargeted) profanity. In this sub-task, there are 2 categories in which the tweet could be classified:

(NOT) Not Offensive - This post does not contain offense or profanity.

(OFF) Offensive - This post contains offensive language or a targeted (veiled or direct) offense. In the annotation, this category includes insults, threats, and posts containing profane language and swear words.

During pre-evaluation period, we trained our models on the train set, and evaluated our different approaches on the trial set. During evaluation period, we trained our models on the train and trial sets, and tested the model on the test set. Table 1 shows the number of tweets per class for English language used in our experiments.

Dataset	NOT	OFF	Total
Train	8840	4400	13,240
Trial	243	77	320
Test	-	-	860

Table 1: Number of tweets per class in OffensEval dataset.

3 Resources

For this subtask A, we used different lexicons that we explain in detail below.

VADER (Valence Aware Dictionary and sEntiment Reasoner) (Gilbert, 2014). The VADER sentiment lexicon is a rule-based sentiment analysis tool. This is sensitive both to the polarity and the intensity of sentiments expressed in social media contexts, and is also generally applicable to sentiment analysis in other domains. VADER has been validated by multiple independent human judges. The tool return four values: positive, negative, neutral and compound. The first three scores represent the proportion of text that falls in these categories. The compound score is computed by summing the valence scores of each word in the lexicon, adjusted according to the rules, and then normalized between -1 (most extreme negative) and +1 (most extreme positive).

Offensive/Profane Word List (von Ahn, 2009). A list of 1,384 English terms (unigrams and bigrams) that could be found offensive. The list contains some words that many people won't find offensive, but it's a good start for anybody wanting to detect offensive or profane terms.

4 System Description

In this section, we describe the systems developed for the subtask A in OffensEval task. During our experiments, scikit-learn machine learning in Python library (Pedregosa et al., 2011) was used for benchmarking. A general scheme of the system can be seen in Figure 1.

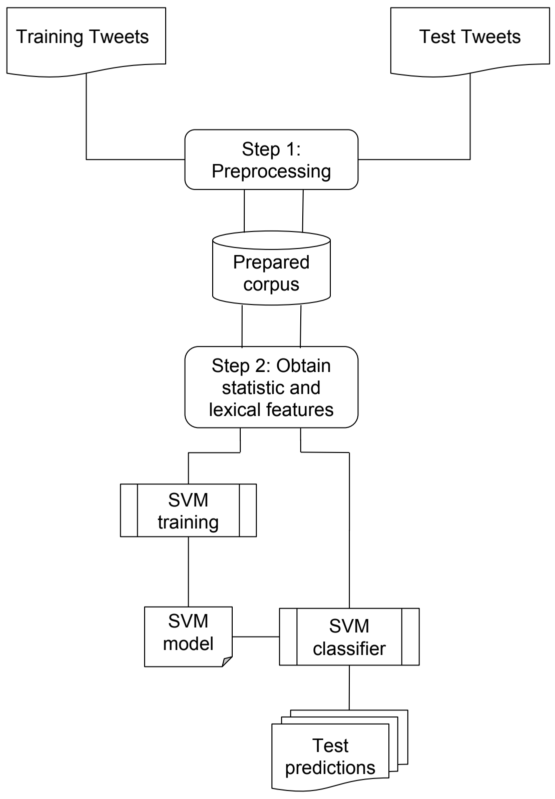


Figure 1: Systems architecture.

4.1 Data Preprocessing

In first place, we preprocessed the corpus of tweets provided by the organizers. We applied the following preprocessing steps: the documents were tokenized using NLTK, the URLs and mentions users are removed and all letters were converted to lower-case.

4.2 Feature Extractor

Converting sentences into feature vectors is a focal task of supervised learning based sentiment analysis. Therefore, the features we chose in our system can be divided into two parts: statistic features and lexical features.

- **Statistic features**. We employed the features that usually perform well in text classification: Term Frequency (TF) taking into account unigrams.
- **Lexical features**. As we explained in Section 3, we used two lexicons to obtain our features in the following way:
    1. **VaderSentiment**. We use the sentiment.vader module[1] provided by the Natural Language Toolkit (NLTK). With this module, we analyze each sentence and we obtained a vector of four scores: negative sentiment, positive sentiment, neutral sentiment and compound polarity.
    2. **Offensive/Profane Word List**. We checked the presence of each word of offensive/profane word list in the tweet and if it exists we assigned 1 as Confidence Value (CV). Then, we summed the CV of all the words finding in the tweet and this value is divided for the total number of words of tweet. As a result, we obtained a parameter that will be used as a feature applied for the classifier.

4.3 Classifier

The concatenation of the features described before are applied for the classification using the SVM algorithm. We selected the Linear SVM formulation, known as C-SVC and the value of the C parameter was 1.0.

5 Analysis of results

During the pre-evaluation phase we carried out several experiments and the best experiment were taken into account for the evaluation phase. The system has been evaluated using the official competition metric, the macro-averaged F1-score. The metric has been computed as follows:

$$\text{Macro-F1} = \frac{2 * \text{Macro-Prec} * \text{Macro-Rec}}{\text{Macro-Prec} + \text{Macro-Rec}} \quad (1)$$

The results of our participation in the subtask A of OffensEval task during the evaluation phase can be seen in Table 2.

[1]https://www.nltk.org/_modules/nltk/sentiment/vader.html

Class	precision	recall	f1-score
NOT	0.81	0.95	0.88
OFF	0.77	0.44	0.56
avg / total	0.8	0.81	**0.79**

Table 2: System test results per class in subtask A of OffensEval task.

User name (ranking)	Macro-F1
pliu19 (1)	0.83
DA-LD-Hildesheim (22)	0.78
fmplaza (68)	**0.72**
gretelliz92 (80)	0.67
AyushS (102)	0.42

Table 3: System Results per participating team in subtask A of OffensEval task.

In relation to our results, it should be noted that we achieve better score in case of the class NOT offensive (F1: 0.88). However, our system is not able to classify well the OFF class (F1: 0.56). This issue may be due to overtraining for the NOT class since as we can see in the Table 1 of Section 2, around 67% of the total tweets belong to that class in the training set in comparison to 33% of OFF class.

With respect to other users, we were ranked in the 68th position as can be seen in Table 3.

6 Conclusions and Future Work

In this paper, we present the system we have developed as part of our participation in SemEval-2019 Task 6: OffensEval: Identifying and Categorizing Offensive Language in Social Media. Specifically, we have participated in subtask A. To solve this task, we have developed a classifier system based on SVM incorporating lexical features from a polarity lexicon and a offensive/profane word list.

Our next study will focus on exploring more features from lexicons because in SemEval-2018 Task 1 (Mohammad et al., 2018), most of the top-performing teams relied on features derived from existing affective lexicons. Also, we will continue working on classifying offensive tweets because today it is a very important task due to the large amount of offensive data generated by users on the Web and we need to prevent the serious consequences it can have on other users.

7 Acknowledgments

This work has been partially supported by Fondo Europeo de Desarrollo Regional (FEDER) and REDES project (TIN2015-65136-C2-1-R) from the Spanish Government.

References

Luis von Ahn. 2009. Offensive/profane word list. *Retrieved June*, 24:2018.

J Cheng. 2007. Report: 80 percent of blogs contain offensive content. *Ars Technica*, 2011.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Eleventh International AAAI Conference on Web and Social Media*.

Aditya Gaydhani, Vikrant Doma, Shrikant Kendre, and Laxmi Bhagwat. 2018. Detecting hate speech and offensive language on twitter using machine learning: An n-gram and tfidf based approach. *arXiv preprint arXiv:1809.08651*.

CJ Hutto Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14). Available at (20/04/16) http://comp. social. gatech. edu/papers/icwsm14. vader. hutto. pdf*.

Sameer Hinduja and Justin W Patchin. 2018. Connecting adolescent suicide to the severity of bullying and cyberbullying. *Journal of School Violence*, pages 1–14.

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 task 1: Affect in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 1–17.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

SSN_NLP at SemEval-2019 Task 6: Offensive Language Identification in Social Media using Traditional and Deep Machine Learning Approaches

D. Thenmozhi, B. Senthil Kumar, Chandrabose Aravindan, S. Srinethe
Department of CSE, SSN College of Engineering, India
{theni_d,senthil,aravindanc}@ssn.edu.in
srinethe16108@cse.ssn.edu.in

Abstract

Offensive language identification (OLI) in user generated text is automatic detection of any profanity, insult, obscenity, racism or vulgarity that degrades an individual or a group. It is helpful for hate speech detection, flame detection and cyber bullying. Due to immense growth of accessibility to social media, OLI helps to avoid abuse and hurts. In this paper, we present deep and traditional machine learning approaches for OLI. In deep learning approach, we have used bi-directional LSTM with different attention mechanisms to build the models and in traditional machine learning, TF-IDF weighting schemes with classifiers namely Multinomial Naive Bayes and Support Vector Machines with Stochastic Gradient Descent optimizer are used for model building. The approaches are evaluated on the OffensEval@SemEval2019 dataset and our team SSN_NLP submitted runs for three tasks of OffensEval shared task. The best runs of SSN_NLP obtained the F1 scores as 0.53, 0.48, 0.3 and the accuracies as 0.63, 0.84 and 0.42 for the tasks A, B and C respectively. Our approaches improved the base line F1 scores by 12%, 26% and 14% for Task A, B and C respectively.

1 Introduction

Offensive language identification (OLI) is a process of detecting offensive language classes (Razavi et al., 2010) such as slurs, homophobia, profanity, extremism, insult, disguise, obscenity, racism or vulgarity that hurts or degrades an individual or a group from user-generated text like social media postings. OLI is useful for several applications such as hate speech detection, flame detection, aggression detection and cyber bullying. Recently, several research work have been reported to identify the offensive languages using social media content. Several workshops such as TA-COS[1], TRAC[2] (Kumar et al., 2018a), Abusive Language Online[3] and GermEval (Wiegand et al., 2018) have been organized recently in this research area. In this line, OffensEval@SemEval2019 (Zampieri et al., 2019b) shared task focuses on identification and categorization of offensive language in social media. It focuses on three subtasks namely offensive language detection, categorization of offensive language and offensive language target identification. Sub_Task_A aims to detect text as offensive (OFF) or not offensive (NOT). Sub_Task_B aims to categorize the offensive type as targeted text (TIN) or untargeted text (UNT). Sub_Task_C focuses on identification of target as individual (IND), group (GRP) or others (OTH). Our team SSN_NLP participated in all the three subtasks.

2 Related Work

Several research work have been reported since 2010 in this research field of hate speech detection (Kwok and Wang, 2013; Burnap and Williams, 2015; Djuric et al., 2015; Davidson et al., 2017; Malmasi and Zampieri, 2018; Schmidt and Wiegand, 2017; Fortuna and Nunes, 2018; ElSherief et al., 2018; Gambäck and Sikdar, 2017; Zhang et al., 2018; Mathur et al., 2018). Schmidt and Wiegand (2017) & Fortuna and Nunes (2018) reviewed the approaches used for hate speech detection. Kwok and Wang (2013) used bag of words and bi-gram features with machine learning approach to classify the tweets as "racist" or "nonracist". Burnap and Williams (2015) developed a supervised algorithm for hateful and antagonistic content in Twitter using voted ensemble meta-

[1] http://ta-cos.org/
[2] https://sites.google.com/view/trac1/home
[3] https://sites.google.com/site/abusivelanguageworkshop2017/

classifier. Djuric et al. (2015) learnt distributed low-dimensional representations of social media comments using neural language models for hate speech detection. Davidson et al. (2017) used n-gram (bigram, unigram, and trigram) features with TF-IDF score along with crowd-sourced hate speech lexicon and employed several classifiers including logistic regression with L1 regularization to separate hate speech from other offensive languages. Malmasi and Zampieri (2018) used n-grams, skip-grams and clustering-based word representations as features with ensemble classifier for hate speech detection. ElSherief et al. (2018) performed linguistic and psycholinguistic analysis to detect the hate speech is either "directed" towards a target, or "generalized" towards a group. Gambäck and Sikdar (2017) used deep learning using CNN models to detect the hate speech as "racism", "sexism", "both" and "non-hate-speech". They used character 4-grams, word vectors based on word2vec, randomly generated word vectors, and word vectors combined with character n-grams as features in their approach. Zhang et al. (2018) used convolution-GRU based deep neural network for detecting hate speech.

Many research work have been carried out in aggression detection (Aroyehun and Gelbukh, 2018; Madisetty and Desarkar, 2018; Raiyani et al., 2018; Kumar et al., 2018b). Aroyehun and Gelbukh (2018) & Raiyani et al. (2018) used LSTM and CNN respectively to detect aggression in text. Kumar et al. (2018b) presented the findings of the shared task on aggression identification which aims to detect different scales of aggression namely "Overtly Aggressive", "Covertly Aggressive", and "Non-aggressive". Madisetty and Desarkar (2018) used CNN, LSTM and Bi-LSTM to detect the above scales of aggression. Waseem et al. (2017) & Park and Fung (2017) presented the methodologies on abusive language identification using deep neural networks.

Research on identifying offensive languages has been focused on non-English languages like German (Wiegand et al., 2018), Hindi (Kumar et al., 2018b), Hinglish: Hindi-English (Mathur et al., 2018), Slovene (Fišer et al., 2017) and Chinese (Su et al., 2017). Wiegand et al. (2018) presented an overview of GermEval shared task on the identification of offensive language that focused on classification of German tweets from Twitter. Kumar et al. (2018b) focused on the shared task to identify aggression on Hindi text. Mathur et al. (2018) applied transfer learning to detect three classes namely "nonoffensive", "abusive" and "hate-speech" from Hindi-English code switched language. Fišer et al. (2017) presented a framework to annotate offensive labels in Slovene. Su et al. (2017) rephrased profanity in Chinese text after detecting them from social media text.

3 Data and Methodology

In our approach, we have used OLID dataset (Zampieri et al., 2019a) given by OffensEval@SemEval2019 shared task. The dataset is given in *.tsv* file format with columns namely, ID, INSTANCE, SUBA, SUBB, SUBC where ID represents the identification number for the tweet, INSTANCE represents the tweets, SUBA consists of the labels namely Offensive (OFF) and Not Offensive (NOT), SUBB consists of the labels namely Targeted Insult and Threats (TIN) and Untargeted (UNT) and SUBC consists of the labels namely Individual (IND), Group (GRP) and Other (OTH). The dataset has 13240 tweets. All the instances are considered for Sub_Task_A. However, we have filtered and considered the data that are labelled with "TIN/UNT" and "IND/GRP/OTH" for Sub_Task_B and Sub_Task_C respectively by ignoring the instances labelled with "NULL". Thus, we have obtained 4400 and 3876 instances for Sub_Task_B and Sub_Task_C respectively. We have preprocessed the data by removing the URLs and the text "@USER" from the tweets. Tweet tokenizer [4] is used to obtain the vocabulary and features for the training data.

We have employed both traditional machine learning and deep learning approaches to identify the offensive language in social media. The models that are implemented for the three sub-tasks are given in Table 1.

In deep learning (DL) approach, the tweets are vectorized using word embeddings and are fed into encoding and decoding processes. Bidirectional LSTMs are used for encoding and decoding processes. We have used 2 layers of LSTM for this. The output is given to softmax layer by incorporating attention wrapper to obtain the OffensEval class labels. We have trained the deep learning models with a batch size 128 and dropout 0.2 for 300 epochs to build the model. We have em-

[4] https://www.nltk.org/

Tasks	Models	Description
Task A	Task_A_DL_NB	Deep learning with Normed Bahdanau attention
	Task_A_DL_SL	Deep learning with Scaled Luong attention
Tak B	Task_B_DL_NB	Deep learning with Normed Bahdanau attention
	Task_B_DL_SL	Deep learning with Scaled Luong attention
	Task_B_TL_MNB	Traditional Machine Learning with Multinomial Naive Bayes
Task C	Task_C_DL_NB	Deep learning with Normed Bahdanau attention
	Task_C_DL_SL	Deep learning with Scaled Luong attention
	Task_C_TL_SVM	Traditional Machine Learning with Support Vector Machine and Stochastic Gradient Descent optimizer

Table 1: Models for the Tasks

ployed two attention mechanisms namely Normed Bahdanau (NB) (Sutskever et al., 2014; Bahdanau et al., 2014) and Scaled Luong (SL) (Luong et al., 2015, 2017) in this approach. These two variations are implemented to predict the class labels for all the three sub tasks. These attention mechanisms help the model to capture the group of input words relevant to the target output label. For example, consider the instance in Task C: “we do not watch any nfl games this guy can shove it in his pie hole”. This instance clearly contains the offensive slang “pie hole” and about watching the “nfl games”. The attention mechanism captures these named entities or group of words and correctly map to the label “GRP”. Also, it is evident from the earlier experiments (Sutskever et al., 2014; Thenmozhi et al., 2018) that bi-directional LSTM with attention mechanism performs better for mapping input sequences to the output sequences.

In traditional learning (TL) approach, the features are extracted from the tokens with minimum count of two. The feature vectors are constructed using TF-IDF scores for the training instances. We have chosen the classifiers namely Multinomial Naive Bayes (MNB) and Support Vector Machine (SVM) with Stochastic Gradient Descent optimizer to build the models for Task B and Task C respectively. These classifiers have been chosen based on the cross validation accuracies. The class labels namely “TIN/UNT” and “IND/GRP/OTH” are predicted for Task B and Task C using the respective models.

4 Results

We have evaluated our models using the test data of OffensEval@SemEval2019 shared task for the three sub tasks. The performance was analyzed using the metrics namely precision, recall, macro-averaged F1 and accuracy. The results of our approaches are presented in Tables 2, 3 and 4 for Task A, Task B and Task C respectively. We have obtained the best results for Task_A_DL_SL, Task_B_DL_NB, Task_C_TL_SVM models for Task A, Task B and Task C respectively.

System	F1 (macro)	Accuracy
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
Task_A_DL_NB (527733)	0.5166	0.614
Task_A_DL_SL (527740)	**0.5341**	**0.6349**

Table 2: Results for Sub-task A.

System	F1 (macro)	Accuracy
All TIN baseline	0.4702	0.8875
All UNT baseline	0.1011	0.1125
Task_B_DL_NB (532649)	**0.4800**	**0.8375**
Task_B_DL_SL (532651)	0.4558	0.8375
Task_B_TL_MNB (532654)	0.4558	0.7792

Table 3: Results for Sub-task B.

System	F1 (macro)	Accuracy
All GRP baseline	0.1787	0.3662
All IND baseline	0.2130	0.4695
All OTH baseline	0.0941	0.1643
Task_C_DL_NB (536200)	0.2462	**0.4507**
Task_C_DL_SL (536201)	0.2663	0.4178
Task_C_TL_SVM (536203)	**0.3001**	0.4178

Table 4: Results for Sub-task C.

The attention mechanism Scaled Luong performs better when more data is available for training. Normed Bahdanau attention mechanism performs better even for a small dataset. However, deep learning gives poor results than traditional learning approach for Task C, because only 3876 instances were considered for model building. The deep learning model could not learn the features appropiately due to less domain knowledge imparted by the smaller data set. Thus, traditional learning performs better with the given data size when compared to deep learning for Task C. The confusion matrix for our best run in the three sub tasks are depicted in Tables 5, 6 and 7. These tables show that the true positive rate of "NOT", "TIN" and "IND" classes are good as the number of samples for those classes are more in training set. Our approaches show improvement over the base line systems for all the three tasks. We have obtained 12% and 14% improvement on F1 and accuracy respectively for Task A when compared with the base line. For Task B, we have obtained 26% and 34% improvement on F1 and accuracy respectively. Also, Task C results have been improved by 14% and 7% for F1 and accuracy when compared to base line results.

	OFF	NOT
OFF	73	147
NOT	167	473

Table 5: Confusion Matrix for Task_A_DL_SL.

	TIN	UNT
TIN	200	26
UNT	13	1

Table 6: Confusion Matrix for Task_B_DL_NB.

	GRP	IND	OTH
GRP	16	26	7
IND	62	71	27
OTH	0	3	2

Table 7: Confusion Matrix for Task_C_TL_SVM.

5 Conclusion

We have implemented both traditional machine learning and deep learning approaches for identifying offensive languages from social media. The approaches are evaluated on OffensEval@SemEval2019 dataset. The given instances are preprocessed and vectorized using word embeddings in deep learning models. We have employed 2 layered bi-directional LSTM with Scaled Luong and Normed Bahdanau attention mechanisms to build the model for all the three sub tasks. The instances are vectorized using TF-IDF score for traditional machine learning models with minimum count two. The classifiers namely Multinomial Naive Bayes and Support Vector Machine with Stochastic Gradient Descent optimizer were employed to build the models for sub tasks B and C. Deep learning with Scaled Luong attention, deep learning with Normed Bahdanau attention, traditional machine learning with SVM give better results for Task A, Task B and Task C respectively. Our models outperform the base line for all the three tasks. The performance may be improved further by incorporating external datasets (Kumar et al., 2018a; Davidson et al., 2017), lexicons and dictionaries.

References

Segun Taofeek Aroyehun and Alexander Gelbukh. 2018. Aggression detection in social media: Using deep neural networks, data augmentation, and pseudo labeling. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 90–97.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Pete Burnap and Matthew L Williams. 2015. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet*, 7(2):223–242.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech

Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pages 29–30. International World Wide Web Conferences Steering Committee.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Darja Fišer, Tomaž Erjavec, and Nikola Ljubešić. 2017. Legal Framework, Dataset and Annotation Schema for Socially Unacceptable On-line Discourse Practices in Slovene. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018a. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Ritesh Kumar, Atul Kr Ojha, Shervin Malmasi, and Marcos Zampieri. 2018b. Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 1–11.

Irene Kwok and Yuzhou Wang. 2013. Locate the hate: Detecting Tweets Against Blacks. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. 2017. Neural machine translation (seq2seq) tutorial. *https://github.com/tensorflow/nmt*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Sreekanth Madisetty and Maunendra Sankar Desarkar. 2018. Aggression detection in social media using deep neural networks. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 120–127.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Puneet Mathur, Rajiv Shah, Ramit Sawhney, and Debanjan Mahata. 2018. Detecting offensive tweets in hindi-english code-switched language. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, pages 18–26.

Ji Ho Park and Pascale Fung. 2017. One-step and two-step classification for abusive language detection on twitter. *arXiv preprint arXiv:1706.01206*.

Kashyap Raiyani, Teresa Gonçalves, Paulo Quaresma, and Vítor Nogueira. 2018. Fully connected neural network with advance preprocessor to identify aggression over facebook and twitter. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*. TRAC-2018-ACL.

Amir H Razavi, Diana Inkpen, Sasha Uritsky, and Stan Matwin. 2010. Offensive language detection using multi-level classification. In *Canadian Conference on Artificial Intelligence*, pages 16–27. Springer.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Huei-Po Su, Chen-Jie Huang, Hao-Tsung Chang, and Chuan-Jie Lin. 2017. Rephrasing Profanity in Chinese Text. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

D Thenmozhi, B Senthil Kumar, and Chandrabose Aravindan. 2018. Ssn_nlp@ iecsil-fire-2018: Deep learning approach to named entity recognition and relation extraction for conversational systems in indian languages. *CEUR*, 2266:187–201.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

Stop PropagHate at SemEval-2019 Tasks 5 and 6: Are abusive language classification results reproducible?

Paula Fortuna[1,2] **Juan Soler-Company**[2] **Sérgio Nunes**[1,3]

(1) INESC TEC and (3) FEUP, University of Porto
Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal
paula.fortuna@fe.up.pt, sergio.nunes@fe.up.pt
(2) Pompeu Fabra University
Carrer de Roc Boronat 138, 08018 Barcelona, Spain
juan.soler@upf.edu

Abstract

This paper summarizes the participation of Stop PropagHate team at SemEval 2019. Our approach is based on replicating one of the most relevant works on the literature, using word embeddings and LSTM. After circumventing some of the problems of the original code, we found poor results when applying it to the HatEval contest (F1=0.45). We think this is due mainly to inconsistencies in the data of this contest. Finally, for the OffensEval the classifier performed well (F1=0.74), proving to have a better performance for offense detection than for hate speech.

1 Introduction

In the last few years, several evaluation tasks in the context of hate speech detection and categorization have been created. Some of these tasks include e.g., EVALITA (Bosco et al., 2018) and TRAC-1 (Kumar et al., 2018). These type of initiatives promote the development of different but comparable solutions for the same problem, within a short period of time, which is an interesting contribution for a research field. In this paper, we describe the participation of team "Stop PropagHate" in the HatEval and OffensEval tasks of SemEval 2019.

The main goal of both tasks is to improve the classification of Hate Speech and Offensive Language. Some of the works in the literature achieve a very competitive performance, e.g. Badjatiya et al. (2017) obtain an F1 score of 0.93 when using deep learning for classifying hate speech in one of the most commonly used baseline datasets (e.g. Waseem (2016)). In this context, we have a specific objective with our approach: we aim to reproduce a state-of-the-art classifier as described in the literature of this topic.

We choose to reproduce the study by Badjatiya et al. (2017), not only because of the good performance of the developed models, but also because in this work the authors published their code. Considering the amount of parameters available for definition and tuning in a machine learning classification pipeline, a precise and extensive definition of an experiment's parameters is not simple and is hardly ever provided. Thus, having the code of the experiment is the best way to understand not only which steps were conducted, but also how those steps were indeed executed. This is a highly cited paper, which can be regarded as an indicator of its relevance in the area.

In this paper, we describe our journey in the process of replication and the results achieved when applying this classifier in both shared tasks. The paper is structured as follows: Section 2 briefly reviews the literature, Section 3 presents our methodology, Section 4 describes the tasks and preliminary experiences with the data, Section 5 shows our official results in the shared tasks, and we report the conclusions of our work in Section 6.

2 Related Work

Previous research in the field of automatic detection of hate speech and offensive language can give us insight on how to approach this problem. Two surveys summarize previous research and conclude that the approaches rely frequently on Machine Learning techniques (Schmidt and Wiegand, 2017; Fortuna and Nunes, 2018). Different methods are used, such as word and character n-grams (Liu and Forss, 2014), perpetrator characteristics (Waseem and Hovy, 2016) or "othering language" (Burnap and Williams, 2016). Word embeddings (Djuric et al., 2015) are often used in this field because they can feed deep learning classification algorithms and obtain high performances. Usually, when traditional Machine

Learning classifiers are used, the most frequent algorithms are SVM (Del Vigna et al., 2017) and Random Forests (Burnap and Williams, 2014), but Deep Learning techniques are quickly gaining ground in the area (Yuan et al., 2016; Gambäck and Sikdar, 2017; Park and Fung, 2017). Different studies proved that deep learning algorithms outperform previous approaches (Mehdad and Tetreault, 2016; Park and Fung, 2017; Badjatiya et al., 2017; Del Vigna et al., 2017; Pitsilis et al., 2018; Founta et al., 2018; Zhang et al., 2018; Gambäck and Sikdar, 2017).

Other sources of solutions are previous shared tasks. For EVALITA, the best performing system achieved an F1 score of 0.83 on Facebook data and 0.80 on Twitter data. The best team tested three different classification models: one based on a linear SVM, another one based on a 1-layer BiLSTM and a 2-layer BiLSTM which exploits multi-task learning with additional data. For TRAC-1, the system that achieved the best performance with a F1 value of 0.64 used an LSTM and resorted to translation as a data augmentation strategy.

During the last 2 years, many articles have been published in this area, and one of the main focal points is to find accurate classifiers for the detection and characterization of hate speech. One main dataset is now used (Waseem, 2016), allowing performance comparison between systems. However, it is still not trivial to compare and reproduce the different approaches. Machine learning classification systems involve a long, complex set of steps and parameters and not every paper gives clear and transparent specifications. A precise specification is fundamental for replicating and improving a system.

With this idea in mind, we tried to replicate a paper with promising results as a baseline for our work. We found a paper which describes several classifiers with good performance and that also provides a GitHub repository for the code of the classifiers (as stated before, the work from Badjatiya et al. (2017). In this paper, the authors propose and use different methods. They investigate three neural network architectures applied to the problem of automatic hate speech detection: CNN, LSTM and FastText. In each one of the methods they initialize the weights with either random embeddings or GloVe embeddings. They use a dataset with messages classified as containing sexist hate speech, racist or none (Waseem, 2016). Additionally, they use 10-fold cross validation. The set of experiments achieving better performance consists in using a deep learning architecture, then taking the weights of the last layer and feeding it into a standard machine learning classifier. More particularly, embeddings learned from LSTM model were combined with gradient boosted decision trees and led to the best performance (F1 score of 0.93).

Regarding our specific approach in this shared task, the main research question of our work concerns if it is possible to replicate the results of the aforementioned paper. After trying to replicate their results, we then apply the approach to the two new datasets provided by the shared tasks.

In the next sections, we present our methodology and approach to these shared tasks.

3 Methodology

For conducting this study, we follow a methodology of 10-fold cross-validation with holdout validation (Chollet, 2017). This consists in dividing the data into two sets. One part of the data is used for cross-validation and parameter tuning with grid search on several classification parameters. The second part of the data is used for estimating the performance of this model when applied to classify new data.

In terms of pipeline, we tried to replicate the study by Badjatiya et al. (2017), and we started by downloading the version of the code[1] in December 2018. We then faced some difficulties that we list here:

- Unspecified versions of Python and of some of the used libraries.
- The authors use the fact that they provide the code as a reason not to specify the parameters in detail.
- The code contains only some of the classifiers described in the paper. The set of classifiers using xgBoost together with deep learning as features were not provided. These are the classifiers with the best performance.
- No validation data is hold out for the model to be tested after the tuning during the cross validation.

[1] https://github.com/pinkeshbadjatiya/twitter-hatespeech

- In the provided code, the 10-fold cross-validation procedure has a bug. With a more detailed analysis of the code we have found out that the method used for classification is train_on_batch from Keras[2], that runs a single gradient update on a single batch of data. Successive calls to this method are done through the 10 iterations of the cross-validation procedure, without instantiating a new model. This means that, during the 10 iterations, the model will successively update the gradient values without resetting it. The effect of using 10-fold cross-validation is then eliminated because only in the first iteration the testing is conducted in data never seen previously by the model. As a consequence of this problem, we can see that the successive values of F1 score found in the 10 iterations increases every time. See Table 1 for the specific F1 score values per cross-validation phase.

CV Iteration	Macro F1
1	0.76
2	0.78
3	0.8
4	0.83
5	0.87
6	0.88
7	0.89
8	0.91
9	0.92
10	0.89
Average	**0.89**

Table 1: F1 score over the different iterations of the cross-validation procedure. Experiment conducted for replicating the paper from Badjatiya et al. (2017).

In order to overcome these limitations, we provide all the information required to replicate the experiment. We use Python 3.6, Keras (Chollet et al., 2018), Gensim (Řehůřek and Sojka, 2010) and Scikit-learn (Pedregosa et al., 2011) as main libraries, and we make available our project and code[3].

The following subsections describe specific indications on how we implement each step performed by our system.

[2]Line 204 at the file `https://github.com/pinkeshbadjatiya/twitter-hatespeech/blob/master/lstm.py`

[3]`https://github.com/paulafortuna/SemEval_2019_public`

3.1 Text pre-processing

In terms of text pre-processing, we remove stop words using Gensim, and punctuation using the default string library. We transform our tweets to lower case.

3.2 Feature extraction

Regarding the features that we use in our experiment, we extract Glove Twitter word embeddings, sentiment and frequencies of words from Hatebase. The last is a set of features developed in our work. In Table 2 we present an overview of the features.

Used features	Dimensions	Abbreviation
Glove twitter word embeddings	200	glove
Sentiment Vader	4	sentiment
Hatebase	2	hatebase

Table 2: Experiment features.

3.2.1 Word embeddings

Regarding the pre-trained word embeddings, we use Twitter Glove pre-trained word embeddings with 200 dimensions. We then use the methods provided by Keras to map each token in the input to an embedding.

3.2.2 Sentiment Features

Another set of features that we use is the sentiment analysis provided by the Vader library (Hutto and Gilbert, 2014). We extract the negative ('neg'), neutral ('neu'), positive ('pos') and compound ('compound') dimensions. Each text is then represented as a 4-dimensional vector with these values.

3.2.3 Hatebase Features

Finally, we use word frequencies from the Hatebase platform (Hatebase, 2019). This platform provides different data regarding hateful words usually connected to hate speech. For each method, we count two set of words:

- Hateful words - corresponds to one or two words and are defined as "terms" in Hatebase (e.g. bitch);
- Hate topic words - corresponds to a definition of the hateful terms and are defined as "hateful meaning" (e.g. a human female). We excluded the stop words and counted for each message if there would be any reference to

words used to explain hatebase terms, so that we could approximate reference to hate related topics.

For every message, we store the frequencies of total hateful words in the text and also the frequencies of hate topic words.

3.3 Classification

Regarding the classifiers we used LSTM and xgBoost.

3.3.1 Deep Learning

For the deep learning model, we used LSTM as implemented in the code from the paper by Badjatiya et al. (2017). This contains an Embedding Layer with the weights from the word embeddings extraction procedure, an addtional LSTM layer with 50 dimensions, and dropouts at the end of both layers. We used Adam as optimizer, binary cross-entropy as loss function, 10 epochs and 128 for batch size. With this model we classify the data into binary classes and we save the last layer before the classification to extract 50 dimensions for giving it as input to the xgBoost algorithm, in a similar manner as described in the paper we are replicating. Additionally, we tested with higher dimensionality, but we find no improvement when we kept the remaining parameters.

3.3.2 xgBoost

We used the gradient boosting algorithm from the Python library xgboost (Chen and Guestrin, 2016). In terms of parameters, we used the default except for the eta and gamma. In this case we conducted a grid search combining several values of both (eta: 0, 0.3, 1; and gamma: 0.1, 1, 10). Additionally, we ran all the possible combinations of the three available sets of features: hatebase words frequencies, sentiment, and weights extracted from the LSTM model.

4 Tasks, systems and results

We conduct different experiments following the procedure described in Section 3.

4.1 Standard Dataset

In our methodology, we use a standard dataset (Waseem, 2016), so that we could compare our results with the original paper we are replicating.

4.1.1 Data

We randomly divided the data into 90% training and 10% testing datasets, having 15,214 messages for training and 1,691 messages for testing.

4.1.2 Results for Tuning and Validation

In Table 3 we present the results of the experiments with the baseline dataset. Some patterns of the results are in accordance with the original study (Badjatiya et al., 2017). Indeed, classifying the data with xgBoost after extracting 50 dimensions with the LSTM brought improvement when compared to directly classify it with LSTM. However, the results presented here are far from the 0.93 reported in the original paper. We obtained an F1 score of 0.72 using cross validation and 0.78 using the test set, when combining LSTM last layer with sentiment, hatebase, and xgBoost as classifier. One explanation for the differences between our results and the cited paper can be the fact that in this experiment we developed and classified hate speech as binary classes and for that we converted the sexism and racism, to a single hate speech class. On the other hand, the original work was conducted with the three original classes of the dataset. Another possible explanation may be the different problems found in the code, mainly the bug in the cross-validation. The results reported in the original paper may be classification models that were not tested in new data, and can be overfitted.

We can also conclude that the sentiment and hatebase features did not work well for our classification tasks either when used alone or together with the 50 dimensions extracted from the LSTM last layer to feed the xgBoost model.

4.2 HatEval (Task 5)

The proposed task (Basile et al., 2019) consists in the detection of Hate Speech targeting immigrants and women in Twitter, using texts in Spanish and English. There are two different tasks but our team participated only in the first. In Task A, the teams predict whether a tweet is hateful or not hateful as a binary classification task. This task is composed of two different subtasks, one in English and another in Spanish. The systems are evaluated and ranked using macro averaged F1 score.

4.2.1 Data

All the data provided for the competition was collected from Twitter and manually annotated via the Figure Eight crowdsourcing platform. The data is organized and especially released for the competition. More specifically, there are two datasets including tweets about hate against women and immigrants, in English and Spanish. The task dataset contains 9,000 messages for training, 1,000 messages for testing during developing phase and 3,000 messages for final testing and evaluation of the different teams.

4.2.2 Results for Tuning and Validation

Our team participated in the Task A for English and, during the model development phase, achieved the results presented in Table 4. We obtained similar results when applying the classifier to this dataset, when compared to the baseline dataset. Again, using the xgBoost to classify and the 50 dimensions of the last layer of LSTM as features, brought improvement. We achieved an F1 score of 0.75 using cross validation and 0.68 using the test set. We noticed that in the baseline the testing results improve, when compared to the cross-validation while when using the HatEval dataset those results decreased.

4.3 OffensEval (Task 6)

In OffensEval (Zampieri et al., 2019b), there are three sub-tasks and one of the main goals is to take into account the type and target of offenses. Our team participated in the Task A, about Offensive language identification. Again, Classification systems in all tasks are evaluated using the macro-averaged F1 score.

4.3.1 Data

The data used for the contest were previously presented in another work (Zampieri et al., 2019a). Participants were allowed to use external resources and other datasets for this task. Our team received 13,240 messages for training, 320 messages for testing during model development phase, and 860 messages for final testing and ranking of the different teams.

4.3.2 Results for Tuning and Validation

Our team participated in Task A and, during the model development phase, achieved the results presented in Table 5. We obtained similar results to the baseline dataset and HatEval contest. Again, using the xgBoost to classify brought improvement when compared to just use LSTM to directly classify the data (F1 score of 0.78 using cross validation and 0.80 using the test set). Additionally, we can see that the classification of Offensive discourse achieved a better performance, when compared to the classification of hate speech in previous tasks. This may indicate that offensive language is easier to identify when compared with hate speech, which is consistent with previous studies (Kumar et al., 2018).

5 Shared Task Results

In Table 6 we present the results of the team "Stop PropagHate" in the two contests. Regarding the HatEval, we conclude that the results drastically dropped in the F1 metric from 0.77 in the testing phase to 0.45 in the contest. We believe that this result is due to the sampling procedure used for building the datasets, we noticed that the training was conducted with a very balanced dataset which is an uncommon situation for hate speech automatic detection. We also find strange that the winning team only achieved 0.65 which is a result much lower than the current state of art (F1 of around 0.80). The low performances of the systems and our drop in score from the development phase indicates that there should be important differences in the evaluation dataset with respect to the training material. However, we checked the proportion of hate speech in both datasets and it is equivalent (around 42%).

For the OffensEval task, we achieved more consistent results in all the phases, with a F1 of 0.74 more similar to the 0.80 from the testing phases. The consistency in this case might indicate that the evaluation set is similar to the training material. The winner of the competition scored 0.83, so we can see that our approach is not far from that performance.

6 Conclusion

In this paper, we entered a shared task in the field of hate speech detection and characterization. Our approach was based on replicating one of the most relevant works on the state-of-the-art literature. One of our initial conclusions was that it was not possible to replicate the study and the results we aimed at. Our main difficulty was the lack of specification of the method. Additionally, the incomplete available code contained a bug that brought doubt on the validity of the reported results in the original paper. This allowed us to see the importance of sharing code in this field. This is the

Features	**Classifier**	**Number of features**	**Parameters**	**CV training F1 macro**	**testing F1 macro**
glove	LSTM	28	batch: 128, epochs: 10	0.66	-
hatebase	xgBoost	2	eta: 0, gamma: 0.1	0.44	0.45
sentiment	xgBoost	4	eta: 0, gamma: 1	0.50	0.49
hatebase, sentiment	xgBoost	6	eta: 0, gamma: 0.1	0.50	0.51
LSTM layer	xgBoost	50	eta: 0, gamma: 0.1	0.71	0.77
LSTM layer, hatebase	xgBoost	52	eta: 0, gamma: 1	0.71	0.78
LSTM layer, sentiment	xgBoost	54	eta: 0, gamma: 0.1	0.71	0.77
LSTM layer, hatebase, sentiment	xgBoost	56	eta: 0, gamma: 1	0.72	0.78

Table 3: Achieved F1 score during cross validation (CV) and testing for the baseline experiments.

Features	**Classifier**	**Number of features**	**Parameters**	**CV training F1 macro**	**testing F1 macro**
glove	LSTM	52	batch: 128, epochs: 10	0.67	-
hatebase	xgBoost	2	eta: 0, gamma: 10	0.59	0.54
sentiment	xgBoost	4	eta: 0, gamma: 1	0.61	0.53
hatebase, sentiment	xgBoost	6	eta: 0, gamma: 1	0.56	0.36
LSTM layer	xgBoost	50	eta: 0, gamma: 0.1	0.74	0.68
LSTM layer, hatebase	xgBoost	52	eta: 0, gamma: 0.1	0.75	0.68
LSTM layer, sentiment	xgBoost	54	eta: 0, gamma: 0.1	0.74	0.66
LSTM layer, hatebase, sentiment	xgBoost	56	eta: 0, gamma: 0.1	0.74	0.67

Table 4: Achieved F1 score during cross validation (CV) and testing for the HatEval experiments.

Features	**Classifier**	**Number of features**	**Parameters**	**CV training F1 macro**	**testing F1 macro**
glove	LSTM	79	batch: 128, epochs: 10	0.74	-
hatebase	xgBoost	2	eta: 0, gamma: 0.1	0.47	0.50
sentiment	xgBoost	4	eta: 0, gamma: 1	0.65	0.68
hatebase, sentiment	xgBoost	6	eta: 0, gamma: 0.1	0.41	0.47
LSTM layer	xgBoost	50	eta: 0, gamma: 1	0.78	0.80
LSTM layer, hatebase	xgBoost	52	eta: 0, gamma: 0.1	0.78	0.80
LSTM layer, sentiment vader	xgBoost	54	eta: 0, gamma: 0.1	0.78	0.80
LSTM layer, hatebase, sentiment	xgBoost	56	eta: 0, gamma: 0.1	0.78	0.80

Table 5: Achieved F1 score during cross validation (CV) and testing for the OffensEval experiments.

Metrics	**HatEval Task A**	**OffensEval Task A**
Model 1 (A)	0.48	0.82
Model 2 (A)	-	0.82
Model 3 (A)	-	0.82
Model 1 (F1)	0.45	0.74
Model 2 (F1)	-	0.74
Model 3 (F1)	-	0.74
Classification #	35	44
Number of teams	67	103
1st place (F1)	0.65	0.83
last place (F1)	0.35	0.17

Table 6: Achieved performance in the shared tasks. Model 1 corresponds to the original classifier from the replicated paper (LSTM 50d + xgBoost). Model 2 corresponds to the same as Model 1 plus adding hatebase features, and finally, Model 3 corresponds to the same as Model 1 plus adding hatebase and sentiment features.

only way to exactly replicate the reported results to then apply the same approach in other scenarios. A posteriori, we received an answer from the authors explaining that the available GitHub repository does not correspond to the final version of the project. Nevertheless, the it remained not updated at the moment of the submission of this paper. Another work could also not replicate this results (Lee et al., 2018).

After circumventing some of the aforementioned problems of the original code, we explained our specific version and used it to enter the shared task. We show that using the same classifier we found poor results when applied to the HatEval contest. Due to the results achieved on the baseline dataset and testing set before contest, we think this is due to inconsistencies between the characteristics of the training set and the final test set. Finally, for the OffensEval we believe that the classifier performed well, and with a better performance for offense detection than for hate speech.

Acknowledgments

This work was partially funded by the Google DNI grant Stop PropagHate.

References

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Cristina Bosco, Felice DellOrletta, Fabio Poletto, Manuela Sanguinetti, and Maurizio Tesconi. 2018. Overview of the EVALITA 2018 Hate Speech Detection Task. In *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Pete Burnap and Matthew L. Williams. 2016. Us and them: identifying cyber hate on Twitter across multiple protected characteristics. *EPJ Data Science*, 5(1):11.

Peter Burnap and Matthew L. Williams. 2014. Hate speech, machine classification and statistical modelling of information flows on Twitter: Interpretation and communication for policy decision making. In *Proceedings of Internet, Policy & Politics*, pages 1–18.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. ACM.

Francois Chollet. 2017. *Deep learning with python*. Manning Publications Co.

François Chollet et al. 2018. Keras: The python deep learning library. *Astrophysics Source Code Library*.

Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate me, hate me not: Hate speech detection on facebook. In *Proceedings of the First Italian Conference on Cybersecurity*, pages 86–95.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web*, pages 29–30. ACM2.

Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Antigoni-Maria Founta, Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Athena Vakali, and Ilias Leontiadis. 2018. A unified deep learning architecture for abuse detection. *arXiv preprint arXiv:1802.00385*.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hatespeech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Hatebase. 2019. Hatebase. Available in `https://www.hatebase.org/`, accessed last time in January 2019.

Clayton J. Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Younghun Lee, Seunghyun Yoon, and Kyomin Jung. 2018. Comparative studies of detecting abusive language on twitter. *arXiv preprint arXiv:1808.10245*.

Shuhua Liu and Thomas Forss. 2014. Combining n-gram based similarity analysis with sentiment analysis in web content classification. In *International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, pages 530–537.

Yashar Mehdad and Joel Tetreault. 2016. Do characters abuse more than words? In *Proceedings of the SIGdial 2016 Conference: The 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303.

Ji Ho Park and Pascale Fung. 2017. One-step and Two-step Classification for Abusive Language Detection on Twitter. In *Proceedings of the First Workshop on Abusive Language Online*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Georgios K. Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. *SocialNLP 2017*, page 1.

Zeerak Waseem. 2016. Are you a racist or am i seeing things? annotator influence on hate speech detection on Twitter. In *Proceedings of the 1st Workshop on Natural Language Processing and Computational Social Science*, pages 138–142.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on Twitter. In *Proceedings of NAACL-HLT*, pages 88–93.

Shuhan Yuan, Xintao Wu, and Yang Xiang. 2016. A two phase deep learning model for identifying discrimination from tweets. In *International Conference on Extending Database Technology*, pages 696–697.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting hate speech on Twitter using a convolution-gru based deep neural network. In *European Semantic Web Conference*, pages 745–760. Springer.

TECHSSN at SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Tweets using Deep Neural Networks

Logesh Balasubramanian, Harshini Sathish Kumar,
Geetika Bandlamudi, Dyaneswaran Sivasankaran,
Rajalakshmi Sivanaiah, Angel Deborah Suseelan,
Sakaya Milton Rajendram, Mirnalinee Thanka Nadar Thanagathai
Department of Computer Science and Engineering
SSN College of Engineering
Chennai 603 110, Tamil Nadu, India
{logesh16054, harshini16040}@cse.ssn.edu.in,
{geetika16032, dyaneswaran16028}@cse.ssn.edu.in,
{rajalakshmis, angeldeborahs}@ssn.edu.in
{miltonrs, mirnalineett}@ssn.edu.in

Abstract

Task 6 of SemEval 2019 involves identifying and categorizing offensive language in social media. The systems developed by TECHSSN team uses multi-level classification techniques. We have developed two systems. In the first system, the first level of classification is done by a multi-branch 2D CNN classifier with Google's pre-trained Word2Vec embedding and the second level of classification by string matching technique supported by offensive and bad words dictionary. The second system uses a multi-branch 1D CNN classifier with Glove pre-trained embedding layer for the first level of classification and string matching for the second level of classification. Input data with a probability of less than 0.70 in the first level are passed on to the second level. The misclassified examples are classified correctly in the second level.

1 Introduction

The growth of social media networks in recent days has been phenomenal and Twitter is no exception. However, this rapid growth of social media also poses a serious challenge of maintaining ethics in social media because of the degradation of moral values in society. Offensive micro tweets are generated on a daily basis targeting a particular person, organization, race, caste, community, religion, gender and so forth. For this reason, our task for the SemEval2019 (Zampieri et al., 2019b) mainly focuses on the detection of offensive language in tweets and classify them.

In Subtask-A, we tried to classify the tweets into two classes, namely offensive and non-offensive. The offensive tweets in the Subtask-A are then categorized in Subtask-B into targeted and untargeted tweets, where targeted tweets are aimed at a specific person, organization, religion or political parties. Further, Subtask-C deals with the fine-grained classification of offensive tweets into three classes viz person, organization and others.

The training dataset provided by the organizers contains 13240 tweets. The given dataset is used as the preliminary dataset to train our model. In addition, Impermium dataset from Kaggle and TRAC dataset are added to improve the accuracy of the model. We have also used a dictionary of offensive words in the second level of classification. Manually classifying the tweets is ambiguous and highly subjective, and is one of the biggest challenges. The mix of colloquial slang in tweets, veiled references, missing data, usage of symbols and emojis are further hurdles that lowered the prediction accuracy (Founta et al., 2018).

2 Related Work

Many researchers in the field of Artificial Intelligence and Natural Language Processing have been working to detect offensive speech in tweets using sentiment analysis. Pang et al. (2002) used a three level classification system with Naive Bayes classifier in the first level, Multinomial Updatable Naive Bayes in the second level and a rule based classifier named DTNB in the third level. The second level of classification increased the accuracy by 7% while the third level results showed an improvement in performance by a 6% rise in accuracy. The results were boosted by the usage of an insulting and abusive language dictionary.

Stammbach et al. (2018) developed an ensemble model of Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and CNN+RNN that gave a macro averaged F1-score of 77.7%, 78.6% and 77.6% respectively on 10-

fold cross validation. It is stated that the correct words generated by the spell-checker did not occur in the embeddings and this might be one of the reasons for the low performance of the model. Che et al. (2017) presented a review of the recent deep neural networks used for text classification and a comparison of word embeddings to Support Vector Machine (SVM) classifiers. It points out that a critical issue with CNN is the restriction of a fixed input size and hence the inability to handle sentences of variable length as input, and therefore, focuses on RNN. It is also mentioned that n-grams with syntactic and semantic information achieve significantly better results than the standard n-grams.

(Le and Mikolov, 2014) uses paragraph vectors instead of Bag-of-Words (BOW) feature representation and reduces the classification error by approximately 39%. Dinakar et al. (2011) and ElSherief et al. (2018) discuss the problem of overlapping classes in target identification. Malmasi and Zampieri (2018) discuss the various challenges in discriminating hate, offensive and non-offensive text using ensemble techniques and stacked generalization meta learning methods.

Razavi et al. (2010) discuss about the multilevel classification for flame detection using complement naive Bayes on first level to select discriminative features and multinomial updatable naive Bayes classifier on second level to enhance the model with new features for adaptive learning. They have used rule-based classifier named DTNB (Decision Table/Naive Bayes hybrid classifier) as the last level to classify the text into Flame/Not.

3 Methodology and Data

The task of classifying offensive tweets is difficult as it needs to discover the intention of the user. Moreover, people who follow the chatting convention use offensive words to express their feelings. The architecture diagram for the offensive text classification is shown in Figure 1.

3.1 Acquiring Datasets

The classifier can make a well-informed decision if we procure and supply more data to it. For good performance, deep learning requires sufficiently large amount of data. Therefore, in addition to the dataset given by Zampieri et al. (2019a), we also compiled a variety of datasets for our tweet classification. We have added the TRAC training dataset (Kumar et al., 2018) consisting of online posts from Facebook, the Impermium Dataset of Kaggle (Impermium, 2013), and used a comprehensive list of known offensive words banned by Google in all their different forms. TRAC dataset is based on multiclass classification (OAG, CAG, NAG). We have considered OAG and CAG class labels as OFF label and NAG as NOT for the subtask A.

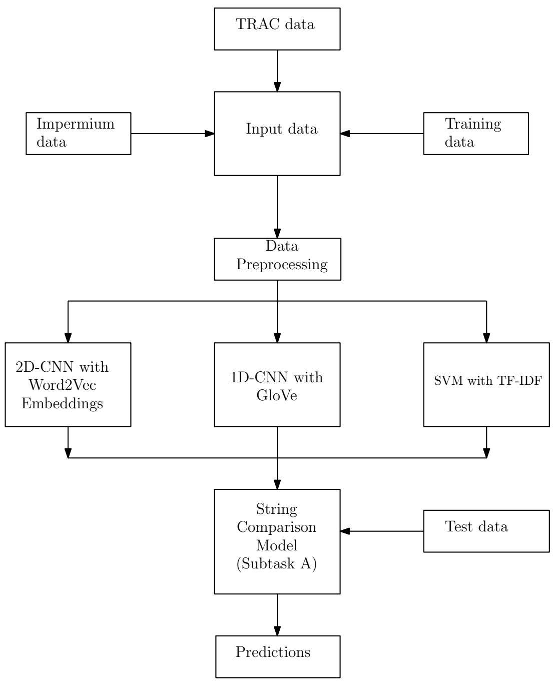


Figure 1: Architecture of Proposed System

3.2 Data Preprocessing

Data preprocessing critical for the success of any machine learning solution. The given dataset shows many signs of irregularities which is a classic signature of any collection of tweets. Normalizing the data involves flattening the dimensions of data into textual form. The dataset is cleaned and processed using functions from NLTK and spacy toolkit.

During preprocessing, we

(a) remove URLs,
(b) annotate emojis, emoticons,
(c) convert uppercase to lowercase,
(d) expand contractions,
(e) remove stopwords,
(f) remove special characters,
(g) remove accented characters,
(h) reduce lengthened words,
(i) lemmatize text, and

(j) remove extra whitespace

Among the steps listed above, step (e) is omitted for subtask-B and subtask-C, since stopwords are significant for target identification.

3.3 Model Description

We classified the text using these three models and compare the results:

1. 2D-CNN with Word2Vec Learned Embeddings
2. 1D-CNN with GloVe
3. SVM with BOW

3.3.1 2D-CNN with Word2Vec Learned Embeddings

Even though it is unconventional to use a two dimensional convolutional network to work on text sequences, it has proved its usefulness quite satisfactorily (Prusa and Khoshgoftaar, 2017). The model is built by taking the pre-trained Google's Word2Vec weights and learn more with the following additional layers.

1. Input layer
2. Embedding layer
3. Convolutional layer with kernel size 2
4. Convolutional layer with kernel size 3
5. Convolutional layer with kernel size 4
6. Respective pooling layers for CNN layers
7. Fully connected dense layer
8. Output layer

We have used an embedding layer which learns the weights of the embedding matrix during training. The bigrams, trigrams and fourgrams of the words are obtained by applying filters and kernels of the right size on the embedding layer output. After applying the filters, a max pooling operation is performed on each CNN layer to scale down the output vectors into dense feature vectors, each of size 100. The three dense vectors are concatenated and flattened into a single dense vector of size 300 in the fully connected layer. The final output is obtained through a dense layer with 2 output units.

The main ideabehind the concatenation of the word grams is to compute n-grams in parallel, add them together and extract as much information as possible from the vectors. This enables the classifier to learn and understand the relationships between the underlying words. The parameters for the model are as set as follows: sequence length of the model is 43, learning rate is set as 0.001 and dropout is set as 0.5, Softmax activation function for output layer and Relu activation function in other layers.

3.3.2 1D-CNN with GloVe

GloVe embeddings with 1 million word vectors of 200 dimension from twitter is used as an alternative method to create the embedding matrix of the embedding layer in the network. We used a conventional convolutional neural network in single dimension and extracted the skip-grams in parallel by using filters of size 2, 3 and 4, each vector of size 100, in three different branches and concatenated them to form one flattened dense vector of size 300. We have used 100 filters and dropout value as 0.2. Softmax activation function is used in output layer and Relu function is used in other layers. To increase the representational power of the neural network, a couple of dense layers are added before the final output layer. This model has fewer trainable parameters, takes less time to train, yet performs on par with 2D-CNN.

3.3.3 SVM with Bag-Of-Words

We implemented a Support Vector Machine using Term Frequency-Inverse Document Frequency (TF-IDF) model and found it to be as good as neural networks. The input document is converted into binary vectors using TF-IDF method. These feature vectors are fed as input to the SVM which uses a linear function as the kernel. The confusion matrix for SVM is shown in Table 1. Linear SVM is found to be better for text classification since most of the text classification problems are linearly separable (Joachims, 1998). For our model, BOW vector size is 98807.

		Actual	
		OFF	NOT
Predicted	OFF	689	618
	NOT	75	1880

Table 1: Linear SVM with TF-IDF model

3.3.4 Logistic Regression and RNN Models

We also trained Logistic Regression (Davidson et al., 2017) and RNN + LSTM (Long Short-Term Memory) models (Pitsilis et al., 2018). Table 2 shows macro average F1-scores for the various models developed. These models do not perform well compared to CNN and SVM. We used 75%

Model Used	F1 (Macro)
1D-CNN with GloVe	**0.751**
2D-CNN with Word2Vec	0.740
Linear SVM with TF-IDF	0.722
RNN+LSTM and GloVe	0.719
Logistic Regression with BoW	0.703
RNN+LSTM and Word2Vec	0.702

Table 2: Comparison of F1-scores of the Models Used

of the instances for training and 25% for testing the accuracy of the models.

The Logistic Regression using count vectorization model predicts more offensive tweets correctly than the previous models specified here. However, since the model could not classify non-offensive tweets effectively, its performance is low. The reason is that the bag of words model using count vectorization does not take the context or the semantics of the tweet into account and hence contextually non-offensive tweets with an offensive word in them are misclassified. Table 3 shows the confusion matrix for logistic regression model.

		Actual	
		OFF	NOT
Predicted	OFF	831	476
	NOT	250	1705

Table 3: Logistic Regression using BoW

3.3.5 Second Level Classification for Subtask-A

Most of the models we have described in the paper classify non-offensive tweets more correctly than offensive ones. To overcome the misclassification of offensive tweets, a second level of classification using string comparison model is done. Tweets predicted offensive with a probability less than 0.70 are passed on to a string comparison model which checks for any occurrence of offensive words. As an aid, a dictionary of 1384 words is constructed with the offensive words banned by Google and a list of bad words. This two level classification system proves to be very effective and increases the performance of the system by 5% in the macro average F1-score. The second level of classification is used to find whether the tweet is offensive or not. There is no need for a second level in subtasks B and C since they do not involve this classification.

4 Results

Table 4 shows the accuracy (Acc) results for subtask A. Three models were developed and tested for the given dataset. 2D-CNN model with enhanced dataset has better F1 score for both offensive and non-offensive tweets in comparison to the 2D-CNN with given dataset alone and 1D-CNN. 1D-CNN has better accuracy than other models. Since the given dataset is biased, some models classify non-offensive tweets more correctly than offensive tweets. Such a model has better accuracy than other models, but lower F1 score.

System	F1 (macro)	Acc
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
2D-CNN (28000 Data)	**0.7382**	0.8058
2D-CNN (13240 Data)	0.7351	0.8035
1D-CNN	0.7281	**0.8174**

Table 4: Results for Subtask A

Table 5 shows the results for subtask B. We developed SVM and 2D-CNN model for subtask B. SVM classifies tweets into targeted and untargeted ones more effectively than 2D-CNN model.

System	F1 (macro)	Accuracy
All TIN baseline	0.4702	0.8875
All UNT baseline	0.1011	0.1125
SVM with TF-IDF	**0.6602**	**0.8208**
2D-CNN	0.5588	0.775

Table 5: Results for Subtask B

Table 6 shows the results for subtask C. SVM with TF-IDF, 1D-CNN with GloVe and 2D-CNN with learned Word2Vec embeddings are used to build models for subtask C. 1D-CNN model has better F1 macro score than other models.

System	F1 (macro)	Accuracy
All GRP baseline	0.1787	0.3662
All IND baseline	0.2130	0.4695
All OTH baseline	0.0941	0.1643
SVM with TF-IDF	0.5095	**0.6808**
1D-CNN	**0.5633**	0.6714
2D-CNN	0.4846	0.6479

Table 6: Results for Subtask C

The confusion matrix for the best model in subtasks A, B and C are shown in Figure 2, 3 and 4 respectively.

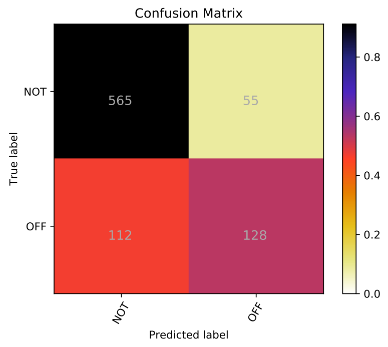


Figure 2: SubTask A: Confusion Matrix for 2D-CNN with Word2Vec embeddings

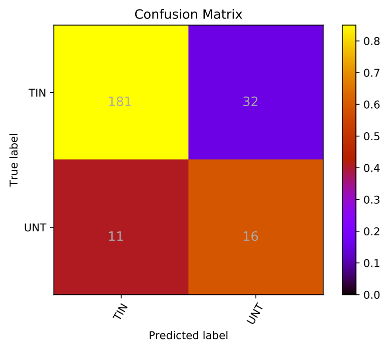


Figure 3: Subtask B: Confusion Matrix for SVM with bag of words model

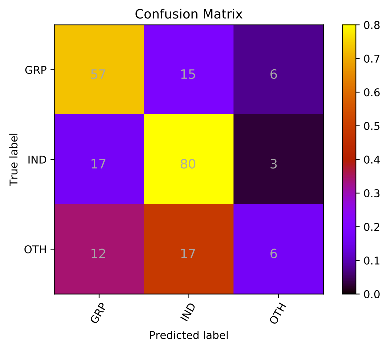


Figure 4: SubTask C: Confusion Matrix for 1D-CNN with GloVe

5 Conclusion

We have built three models for the tasks of offensive language detection and classification in tweets. The models are 2D-CNN with Word2Vec learned embeddings, 1D-CNN with GloVe and SVM with TF-IDF. All the models use data preprocessed with NLTK, which we think is an important factor for improved accuracy. In addition, we also made use of a dictionary of offensive and banned words for a second level of classification.

Meaning of a tweet varies with an individual's perception and cannot be judged by simple conventional models. This is one reason for the reduced precision of classification. The concepts of irony, sarcasm, humor and other tones of a conversation are too intuitive and implicit for the models to detect them accurately. We intend to investigate further by adding multiple hidden layers and building complex network structure which will, in parallel, look for the tell-tale signs of the target tone of the tweets.

1D-CNN model achieved less F1-score in the target identification (subtask C) than in subtasks A and B, due to smaller dataset, which can improved by augmenting the dataset.

References

Hao Che, Susan McKeever, and Sarah Jane Delany. 2017. Abusive text detection using neural networks. In *AICS Conference, Dublin Institute of Technology*.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Karthik Dinakar, Roi Reichart, and Henry Lieberman. 2011. Modeling the detection of textual cyberbullying. In *The Social Mobile Web*, pages 11–17.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large Scale Crowdsourcing and Characterization of Twitter Abusive Behavior. *arXiv preprint arXiv:1802.00393*.

Impermium. 2013. Detecting Insults in Social Commentary. `https://www.kaggle.com/c/detecting-insults-in-social-comme\ntary/data/`.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in discriminating profanity from hate speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30(2):187–202.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Georgios K Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*.

Joseph D Prusa and Taghi M Khoshgoftaar. 2017. Deep neural network architecture for character-level learning on short text. In *The Thirtieth International Flairs Conference*.

Amir H Razavi, Diana Inkpen, Sasha Uritsky, and Stan Matwin. 2010. Offensive language detection using multi-level classification. In *Canadian Conference on Artificial Intelligence*, pages 16–27. Springer.

Dominik Stammbach, Azin Zahraei, Polina Stadnikova, and Dietrich Klakow. 2018. Offensive language detection with neural networks for germeval task 2018. In *14th Conference on Natural Language Processing KONVENS 2018*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

The Titans at SemEval-2019 Task 6: Offensive Language Identification, Categorization and Target Identification

Avishek Garain
Computer Science and Engineering
Jadavpur University, Kolkata
avishekgarain@gmail.com

Arpan Basu
Computer Science and Engineering
Jadavpur University, Kolkata
arpan0123@gmail.com

Abstract

This system paper is a description of the system submitted to "SemEval-2019 Task 6", where we had to detect offensive language in Twitter. There were two specific target audiences, immigrants and women. The language of the tweets was English. We were required to first detect whether a tweet contains offensive content, and then we had to find out whether the tweet was targeted against some individual, group or other entity. Finally we were required to classify the targeted audience.

1 Introduction

Offensive language is pervasive in social media. Individuals frequently take advantage of the perceived anonymity of computer-mediated communication, using this to engage in behavior that many of them would not consider in real life. Online communities, social media platforms, and technology companies have been investing heavily in ways to cope with offensive language to prevent abusive behavior in social media.

One of the most effective strategies for tackling this problem is to use computational methods to identify offense in user-generated content (e.g. posts, comments, microblogs, etc.). This topic has attracted significant attention in recent years of various Natural Language analysts.

The SemEval 2019 task 6 (Zampieri et al., 2019b) was a classification task where we were required to classify a tweet, as hate speech or otherwise. However, there were some additional challenges presented, which involved automatic categorization of offense target types and the specific detection of the target audience, namely, women or immigrants.

The task was divided into three parts. In the first subtask our system categorized the instances into `OFF` and `NOT`. In the second subtask our system categorized instances into `TIN` and `UNT` while in the third subtask systems should categorize instances into `IND`, `GRP`, and `OTH`.

To solve the task in hand we built a bidirectional LSTM based neural network for prediction of the classes present in the provided dataset.

The paper has been organized as follows. Section 2 describes a brief survey on the relevant work done in this field. Section 3 describes the data, on which, the task was performed. The methodology followed is described in Section 4. This is followed by the results and concluding remarks in Section 5 and 6 respectively.

2 Related Work

Papers published in the last two years include the surveys by Schmidt and Wiegand (2017) and Fortuna and Nunes (2018). The paper by Davidson et al. (2017) presenting the Hate Speech Detection dataset were used in (Malmasi and Zampieri, 2017) and a few other recent papers such as (ElSherief et al., 2018; Gambäck and Sikdar, 2017; Zhang et al., 2018).

A proposal of typology of abusive language sub-tasks is presented in (Waseem et al., 2017). For studies on languages other than English see work by Su et al. (2017) on Chinese and Fišer et al. (2017) on Slovene. Finally, for recent discussion on identifying profanity vs. hate speech see the work by Malmasi and Zampieri (2018). This work highlighted the challenges of distinguishing between profanity, and threatening language which may not actually contain profane language.

Previous editions of related workshops are TA-COS[1], Abusive Language Online[2], and TRAC[3] and related shared tasks such as GermEval (Wiegand et al., 2018) and TRAC (Kumar et al., 2018).

[1]http://ta-cos.org/
[2]https://sites.google.com/site/abusivelanguageworkshop2017/
[3]https://sites.google.com/view/trac1/home

3 Data

The dataset that was used to train the model is the OLID dataset (Zampieri et al., 2019a). It was collected from Twitter; the data being retrieved the data using the Twitter API by searching for keywords and constructions that are often included in offensive messages. The vast majority of content on Twitter is not offensive so different strategies were tried to keep a reasonable number of tweets in the offensive class amounting to around 30% of the dataset.

The dataset provided consisted of tweets in their original form along with the corresponding labels. Subtask A consisted of the labels `OFF` and `NOT`; subtask B consisted of the labels `TIN` and `UNT`; and finally subtask C consisted of the labels `IND`, `GRP` and `OTH`.

Label	**Meaning**
OFF	Tweet containing offensive language
NOT	Tweet not containing offensive language
TIN	Tweet containing profanity and targeted against individual/group/others
UNT	Tweet with profanity, but non-targeted
IND	Offensive tweet targeting an individual
GRP	Offensive tweet targeting a group
OTH	Offensive tweet targeting neither group or individual

Table 1: Meaning of the labels used in the dataset

The dataset had 14100 instances which were divided into 13240 training data instances and 860 test data instances.

A	**B**	**C**	**Train**	**Test**	**Total**
OFF	TIN	IND	2407	100	2507
OFF	TIN	OTH	395	35	430
OFF	TIN	GRP	1074	78	1152
OFF	UNT	-	524	27	551
NOT	-	-	8840	620	9460
All			13240	860	14100

Table 2: Distribution of the labels in the dataset

4 Methodology

Our approach was to convert the tweet into a sequence of words and then run a neural-network based algorithm on the processed tweet.

The first stage in our pipeline was to preprocess the tweet. This consisted of the following steps:

1. Removing mentions
2. Removing punctuation
3. Removing URLs
4. Contracting white space
5. Extracting words from hash tags

The last step consists of taking advantage of the Pascal Casing of hash tags (e.g. `#PascalCasing`). A simple regex can extract all words; we ignore a few errors that arise in this procedure. This extraction results in better performance mainly because words in hash tags, to some extent, may convey sentiments of hate. They play an important role during the model-training stage.

We treat the tweet as a sequence of words with interdependence among various words contributing to its meaning. Hence we use an bidirectional LSTM based approach to capture information from both the past and future context.

Our model is a neural-network based model. First, the input tweet is passed through an embedding layer which transforms the tweet into a 128 length vector. The embedding layer learns the word embeddings from the input tweets. This is followed by two bidirectional LSTM layers containing 64 units each. This is followed by the final output layer of neurons with softmax activation, each neuron predicting a label as present in the dataset. For subtasks 1 and 2, it contains 2 neurons for predicting `OFF/NOT` and `TIN/UNT` respectively; for subtask 3 it contains 3 neurons for predicting `IND/GRP/OTH`. Between the LSTM and output layers, we add dropout with a rate of 0.5 as a regularizer. The model is trained using the Adam optimization algorithm with a learning rate of 0.0005 and using crossentropy as the loss.

We note that the dataset is highly skewed in nature. If trained on the entire training dataset without any validation, the model tends to completely overfit to the class with higher frequency as it leads to a higher accuracy score.

To overcome this problem, we took some measures. Firstly, the training data was split into two parts; one for training and one for validation comprising 70 % and 30 % of the dataset respectively. The training was stopped when two consecutive epochs increased the measured loss function value for the validation set.

Secondly, class weights were assigned to the different classes present in the data. The weights were approximately chosen to be proportional to the inverse of the respective frequencies of the classes. Intuitively, the model now gives equal weight to the skewed classes and this penalizes tendencies to overfit to the data.

In general, we took 0.5 as the boundary between predictions of 0 and 1 — essentially rounding the predicted values. However, for subtask B, we try different values for this parameter (`thresh`) to achieve better results. Values less than `thresh` are converted to 0 while the remaining values are converted to 1.

5 Results

We have included the automatically generated tables with our results. We have also included some baselines generated by assigning the same labels for all instances. For example, "All OFF" in subtask A represents the performance of a system that labels everything as offensive. We have used this for comparison.

We have also added the relevant confusion matrices that were provided together with the results.

System	F1 (macro)	Accuracy
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
BiLSTM	0.4650	0.5651

Table 3: Sub-task A, garain CodaLab 528038 (Bi-Directional LSTM)

System	F1 (macro)	Accuracy
All TIN baseline	0.4702	0.8875
All UNT baseline	0.1011	0.1125
BiLSTM(*)	0.4702	0.8875
BiLSTM(thresh=0.50)	0.5733	0.9
BiLSTM(thresh=0.40)	**0.5796**	**0.8833**

Table 4: Sub-task B, garain CodaLab 533103 (Bi-Directional LSTM threshold = 0.40)
* - class weights not used

System	F1 (macro)	Accuracy
All GRP baseline	0.1787	0.3662
All IND baseline	0.2130	0.4695
All OTH baseline	0.0941	0.1643
BiLSTM	0.3262	0.4601

Table 5: Sub-task C, garain CodaLab 535813 (Bi-Directional LSTM)

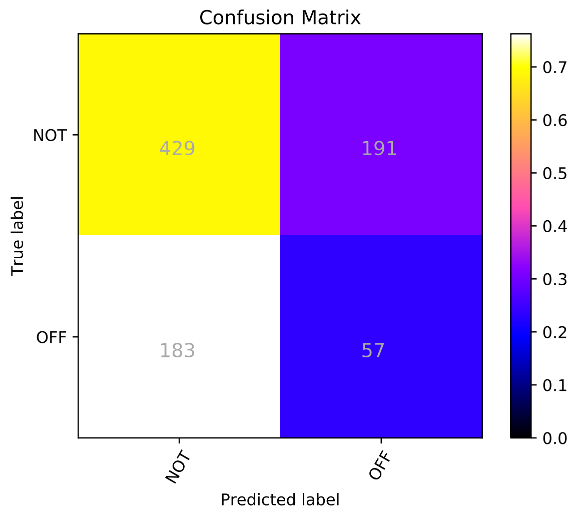


Figure 1: Sub-task A, garain CodaLab 528038 (Bi-Directional LSTM)

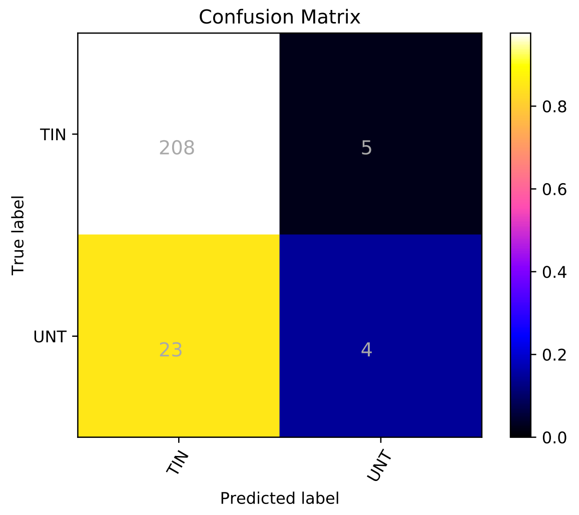


Figure 2: Sub-task B, garain CodaLab 533103 (Bi-Directional LSTM threshold = 0.4)

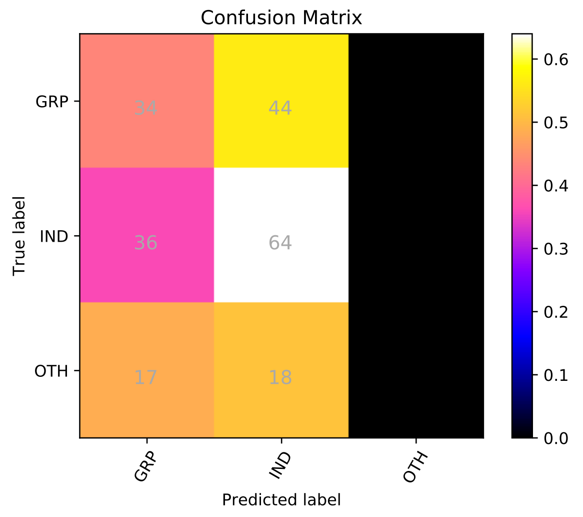


Figure 3: Sub-task C, garain CodaLab 535813 (Bi-Directional LSTM)

6 Conclusion

Here we have presented a model which performs satisfactorily in the given tasks. The model is based on a simple architecture. There is scope for improvement by including more features (like those removed in the preprocessing step) to increase performance. Another drawback of the model is that it does not use any external data other than the dataset provided which may lead to poor results based on the modest size of the data. Related domain knowledge may be exploited to obtain better results.

References

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Darja Fišer, Tomaž Erjavec, and Nikola Ljubešić. 2017. Legal Framework, Dataset and Annotation Schema for Socially Unacceptable On-line Discourse Practices in Slovene. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 467–472.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Huei-Po Su, Chen-Jie Huang, Hao-Tsung Chang, and Chuan-Jie Lin. 2017. Rephrasing Profanity in Chinese Text. In *Proceedings of the Workshop Workshop on Abusive Language Online (ALW)*, Vancouver, Canada.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

TüKaSt at SemEval-2019 Task 6: Something Old, Something Neu(ral): Traditional and Neural Approaches to Offensive Text Classification

Madeeswaran Kannan **Lukas Stein**
Department of Linguistics
University of Tübingen, Germany
{mkannan,lstein}@sfs.uni-tuebingen.de

Abstract

We describe our system (TüKaSt) submitted for Task 6: Offensive Language Classification, at SemEval 2019. We developed multiple SVM classifier models that used sentence-level dense vector representations of tweets enriched with sentiment information and term weighting. Our best results achieved F1 scores of 0.734, 0.660 and 0.465 in the first, second and third sub-tasks respectively. We also describe a neural network model that was developed in parallel but not used during evaluation due to time constraints.

1 Introduction

We live in a day and age where social media pervades through every aspect of lives. Constant growth of social media platforms and rapid exposure to them are changing how human communication is perceived and working (Sticca and Perren, 2013). As much as social media and the general web help its users stay connected and informed, misuse of these channels grows with them. As long as anonymity continues to play a central role in online interactions, one must contend with individuals and entities who feel empowered to behave aggressively for that very reason (Burnap and Williams, 2015). Recent events have forced social media platforms to take a renewed interest in limiting the negative influence of such users for commercial and practical reasons. However, the sheer size of data on social media makes it impossible to manually observe, thereby calling the creation of systems that automatically detect potentially offensive content.

The SemEval 2019 - OffensEval Shared Task (Zampieri et al., 2019b) is aiming exactly at that need, calling for system able to detect offensive language in social media data. The task is split into three sub-tasks: Detecting offensive language, determining whether it is offensive directly towards a target, and target specification. This paper follows our approach of applying both traditional and more sophisticated neural models to the task of offensive text classification.

2 Preprocessing

All tweets were converted to lowercase and tokenised. Stop-words were removed unconditionally, but hashtags and user mentions were removed during the training of certain model variants.

3 Training Data

The SVM models were trained and evaluated exclusively on the training data provided by Zampieri et al. (2019a) with ten-fold cross-validation. For the RNN models, we also used the trail data that was provided before the start of the training phase. The samples were shuffled before each training run, and a 80-20 split was performed to generate training and validation sets.

300-dimensional, pre-trained FastText embeddings (Mikolov et al., 2018) were used to train the SVM models. The FastText embeddings were chosen due to their subword-information (Bojanowski et al., 2017), making it easier to deal with social media data, which is prone to contain spelling errors and abbreviations. 100-dimensional Glove vectors trained on the Twitter corpus (Pennington et al., 2014) were used to initialise the word embeddings in the RNN models.

For sentiment data, we used the Vader sentiment lexicon (Gilbert, 2014) to retrieve polarity scores for each word in the vocabulary. Scores range from -4 (extremely negative) to +4 (extremely positive). A neutral score of zero was assigned to out-of-vocabulary words.

4 Models

4.1 Support Vector Machines Classifier

Support Vector Machines (SVMs) have been used successfully in many classification problems concerning natural language (Aggarwal and Zhai, 2012). In the proceedings of the First Workshop on Trolling, Aggression and Cyberbullying, Kumar et al. (2018) report on several teams using SVMs to identify aggressive texts. Moreover, SVMs with traditional features can still outperform neural networks in natural language tasks (Medvedeva et al., 2017; Çöltekin and Rama, 2017, 2018). However, traditional features like TF-IDF, character and word n-grams, bag of words/n-grams, and sentiment lookups dominate the majority of models used for identifying aggressive language.

Contrasting the use of traditional features used in natural language processing is the prominent research on neural networks using dense vector representations (word embeddings) as input. Since embeddings are able to capture syntactic and semantic features and represent them in a distributed manner, it makes them perfect for language modelling (Bojanowski et al., 2017). We chose to experiment with embeddings in SVMs for this classification task in order to keep pre-processing and feature-engineering to a minimum. Moreover, we wanted to see how the SVM could improve with the dense representations over sparse representations. However, we also experimented with combinations of continuous features and traditional features in some of our models.

The SVM models' dense sentence representation was composed by summing the respective embeddings of every word in the tweet and normalising over the length of the tweet. We chose not to weight them by their TF-IDF scores to achieve an even combination of all constituents. Mikolov et al. (2013) report that adding vectors in a linear fashion yields meaningful phrase representations.

For the first of three SVM models (**combined_fixed**), we created vectors for the TF-IDF and the retrieved sentiment values in the following manner: The TF-IDF vectors were constructed with length $|V|$, where V is the vocabulary constructed from the corpus and each dimension is corresponding to a specific word in the model's vocabulary. The sentiment vector was constructed with length $|V'|$, where V' corresponds to the Vader lexicon (Gilbert, 2014) and each dimension corresponds to a specific word in the lexicon. Both vectors were initialised with zeros and respective dimensions were replaced with the retrieved/computed values. Both vectors were then appended to the dense sentence representation.

The second model (**combined_positioned**) was trained with a word-order sensitive representation of the aforementioned features. Both the TF-IDF vector and the sentiment vectors were constructed with length $|word\_count(t)|$, where t is the longest tweet in the trainings set and where each the dimension corresponds to the word encountered at the respective index in the tweet. The third and final SVM model was only trained on the composed sentence embeddings (**emb_only**).

4.2 Recurrent Neural Classifier

In addition to the SVM classifier, we parallelly trained a recurrent neural classifier using both Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (Cho et al., 2014) cells.

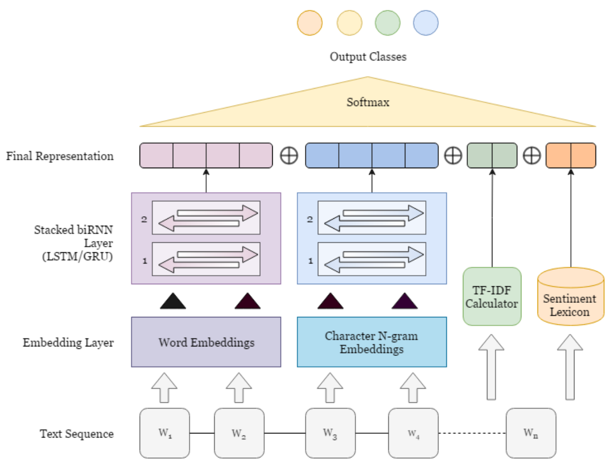


Figure 1: RNN Classifier Architecture.

The architecture of our model is illustrated in figure 1. It accepts token and character n-gram sequences as inputs. The embedding layer consists of individual embedding matrices for each input type. The word embedding matrix is initialised with pre-trained word embeddings while the character embedding matrix is randomly initialised, both of which are subsequently learned as parameters of the model during the training phase.

The embedding sequence is used as the input to a stacked bi-directional RNN layer. We used both LSTM (with peepholes) and GRU cells for the individual units in each RNN layer (trained as separate models). Dropout is additionally added to each layer during the training phase. The hidden states of the forward and backward RNNs of the top-most layer are concatenated to produce the sentential representation of the sequence. This is performed for both types of input sequences.

In addition to the output of the biRNN layers, we calculate TF-IDF (Term Frequency - Inverse Document Frequency) scores for all words in the corpus. This lets us construct a fixed-length dense vector of normalised TF-IDF values for each tweet. Similarly, we use a sentiment lexicon to lookup human-assigned polarity scores of individual words and construct a second fixed-length vector with sentiment information for each tweet.

The outputs of the above components are concatenated into a single, final representation. We expect the TF-IDF and sentiment vectors to encode relevant information about the importance of discriminating words in the input sequence. This final representation is used as the logits for a SoftMax layer that outputs the predicted label of the tweet.

5 Parameters & Training

We used *sklearn-kittext* (Pedregosa et al., 2011) to build our SVM models. For both Sub-task A and B, a binary classifier was trained and for Sub-task C a one-versus-one, multi-class classifier was trained on the three different labels. All three models use sklearn's default RBF-kernel with `C`=2 and `gamma`=0. The final hyper-parameters used by the neural classifier are listed in table 7. Training was performed with early stopping.

During our initial tests, the neural network classifier particularly had trouble generalising to the training data. We found that the class imbalance in the data caused the classifier to be biased towards the majority class and to over-fit the training data. To mitigate this, we weighted the output of the loss function with the ratios of the different classes in each dataset and performed L2 regularisation on the losses of each mini-batch.

6 Results

The results that follow are those only those of the SVM models evaluated during the evaluation phase of the OffenEval task (Zampieri et al., 2019b). Since we were unable to train the RNN classifier in time for the official evaluation phase, we will instead be reporting the scores of the classifier on the validation set (mean and best macro-F1 scores and their corresponding accuracies) in section A.

The model using the sentence embedding on its own is denoted by **emb_only**, the combination of sentence embedding and multiple-hot TF-IDF and sentiment vectors by **combined_fixed** and the position sensitive combination of the vectors by **combined_positioned**.

For Sub-task B and C, all three models were submitted and for sub-task A, only the **emb_only** and the **combined_fixed** were submitted.

System	F1 (macro)	Accuracy
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
combined_fixed	**0.7340**	**0.7942**
emb_only	0.7127	0.7849

Table 1: Results for Sub-task A.

The results for Sub-task A (Table 1) reveal a slight improvement for adding traditional features to the dense sentence representation. This improvement is likely to be due to the sentiment vector, whose values should be very discriminative for tweets containing offensive language. Nevertheless, this information should be encoded in the embeddings as well.

For Sub-task B (Table 2) and Sub-task C, the **emb_only** model actually outperforms the combined vectors. This might be due to the fact that tasks are more fine-grained than Task A and these nuances are captured by the word embeddings, while the sentiment and TF-IDF vectors end up contributing more to the noise than information in the signal. Nevertheless, the SVM seems to be able to handle the data from the composed sentence embeddings quite well.

System	**F1 (macro)**	**Accuracy**
All TIN baseline	0.4702	**0.8875**
All UNT baseline	0.1011	0.1125
combined_positioned	0.6470	0.8
combined_fixed	0.4702	**0.8875**
emb_only	**0.6602**	0.8208

Table 2: Results for Sub-task B.

System	**F1 (macro)**	**Accuracy**
All GRP baseline	0.1787	0.3662
All IND baseline	0.2130	0.4695
All OTH baseline	0.0941	0.1643
combined_positioned	0.4421	**0.5305**
combined_fixed	0.4213	0.4695
emb_only	**0.4652**	0.5164

Table 3: Results for Sub-task C.

The results obtained for Sub-task B reveal that the model trained on **combined_fixed** performs remarkably badly when evaluated on the F1 score. Moreover, the accordance on Accuracy with the All-TIN-Baseline indicates that our model over-fits the training data and classifies all samples as being targeted, which is actually the case. This problem and general performance could probably be improved by spending more time on hyper-parameter training.

The per-class scores of the best performing model in each sub-task are listed in tables 4-6.

	Precision	Recall	F1-score	Samples
NOT	0.8413	0.8806	0.8605	620
OFF	0.6493	0.5708	0.6075	240

Table 4: Per-class performance in Sub-task A, SVM model *combined_fixed*

	Precision	Recall	F1-score	Samples
TIN	0.9427	0.8498	0.8938	213
UNT	0.3333	0.5926	0.4267	27

Table 5: Per-class performance in Sub-task B, SVM model *emb_only*

	Precision	Recall	F1-score	Samples
GRP	0.5306	0.6667	0.5909	78
IND	0.7424	0.4900	0.5904	100
OTH	0.1837	0.2571	0.2143	35

Table 6: Per-class performance in Sub-task C, SVM model *emb_only*

7 Conclusion

In this paper, we demonstrate the use of both Support Vector Machine models and Recurrent Neural models to classify offensive tweets. We show how sentential representations derived from word and character n-gram embeddings can be enriched by including term salience and sentiment information for certain tasks. For more fine-grained tasks, dense vector representations of sentences work well with SVM classifiers. While our results show that traditional methods still outperform neural network models, there is much room for improvement. Future work could investigate the use of more sophisticated mechanisms like attention to potentially increase performance even further.

References

Charu C Aggarwal and ChengXiang Zhai. 2012. A survey of text classification algorithms. In *Mining text data*, pages 163–222. Springer.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Pete Burnap and Matthew L Williams. 2015. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet*, 7(2):223–242.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Çağrı Çöltekin and Taraka Rama. 2017. Tübingen system in vardial 2017 shared task: Experiments with language identification and cross-lingual parsing. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 146–155.

Çağrı Çöltekin and Taraka Rama. 2018. Tübingen-oslo at semeval-2018 task 2: Svms perform better than rnns in emoji prediction. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 34–38.

CJ Hutto Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14). Available at (20/04/16) http://comp. social. gatech. edu/papers/icwsm14. vader. hutto. pdf.*

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Maria Medvedeva, Martin Kroon, and Barbara Plank. 2017. When sparse traditional models outperform dense neural networks: the curious case of discriminating between similar languages. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 156–163.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Fabio Sticca and Sonja Perren. 2013. Is Cyberbullying Worse than Traditional Bullying? Examining the Differential Roles of Medium, Publicity, and Anonymity for the Perceived Severity of Bullying. *Journal of Youth and Adolescence*, (4):739–750.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

A Appendix

Epochs	50
Batch size	512
L2 Beta	0.001
RNN per-layer dropout	0.15
Character n-gram size	3
RNN output dimension	100
Auxiliary vector dimension	30

Table 7: RNN Model Hyper-parameters

A.1 Recurrent Neural Classifier Validation Results

The *-hm* suffix denotes the models that were trained on tweets from which hash tags and user mentions were removed. *-aux* denotes the models that did not use the fixed-length TF-IDF and sentiment vectors in the final representation. In all other models, both of the aforementioned features were preserved. We also report the validation results of the SVM models for comparison.

In Sub-task A (Table 8), the full-LSTM model with TF-IDF and sentiment vectors scored best, followed very closely by the model without hashtags or user mentions. Removing the auxiliary vectors resulted in a net decrease in performance, which shows that sentiment and [term] salience-related information do indeed help the model learn and generalise better. Interestingly, removing hash tags/mentions along with the auxiliary vectors increased performance relative to the previous case. The GRU scored the lowest; this could potentially be attributed to its weakness with long-distance dependencies.

The full-LSTM model continued to outperform the other models in Sub-task B (Table 9) as well. Removing hash tags and user mentions caused a more substantial drop in performance compared to the first sub-task. This follows logically as the given task is to identify targetted tweets; removing information that is intrinsically indicative of the same results in a worse-performing model. Removing the auxiliary vectors causes performance to drop further, further corroborating their informativity. And as before, the GRU finished in the last place.

Sub-Task C (Table 10) saw a reversal of roles between the GRU and full-LSTM models. Between the different variants of the LSTM-based models, the general trend seen in Sub-task A re-emerged. It must, however, be noted that all models performed poorly at this sub-task. We conjecture that this is partly due to the limited amount of training data available for this sub-task. It is also likely that the chosen architecture of the classifier is not sophisticated enough to model the non-linearities in the training data.

	Mean		Best	
System	**F1 (macro)**	**Accuracy**	**F1 (macro)**	**Accuracy**
SVM combined_positioned	**0.7215**	**0.7540**	**0.7441**	**0.7751**
SVM combined_fixed	0.6171	0.7045	0.6321	0.7177
SVM emb_only	0.7160	0.7483	0.7373	0.7691
GRU	0.6240	0.6584	0.6442	0.6941
LSTM	0.6958	0.7332	0.7121	0.7629
LSTM -hm	0.6941	0.7360	0.7126	0.7488
LSTM -aux	0.6830	0.7361	0.6873	0.7559
LSTM -hm -aux	0.6852	0.7166	0.6997	0.7309

Table 8: Validation results for Sub-task A.

	Mean		Best	
System	**F1 (macro)**	**Accuracy**	**F1 (macro)**	**Accuracy**
SVM combined_positioned	0.5877	0.8117	0.6341	0.8616
SVM combined_fixed	0.4677	**0.8791**	0.4737	**0.9002**
SVM emb_only	0.6128	0.8129	**0.6676**	0.8594
GRU	0.5949	0.6350	0.6129	0.6578
LSTM	**0.6527**	0.7065	0.6642	0.7039
LSTM -hm	0.6322	0.6880	0.6372	0.6859
LSTM -aux	0.6267	0.6696	0.6527	0.6809
LSTM -hm -aux	0.6258	0.6528	0.6655	0.6973

Table 9: Validation results for Sub-task B.

	Mean		Best	
System	**F1 (macro)**	**Accuracy**	**F1 (macro)**	**Accuracy**
SVM combined_positioned	0.4003	0.5788	0.4934	0.6340
SVM combined_fixed	0.4919	0.6129	0.5163	0.6417
SVM emb_only	**0.5086**	0.6273	**0.5672**	0.6494
GRU	0.2953	0.5748	0.3109	0.5746
LSTM	0.2878	0.6045	0.2961	0.6164
LSTM -hm	0.2868	0.6365	0.3390	0.6152
LSTM -aux	0.2586	**0.6432**	0.2637	**0.6762**
LSTM -hm -aux	0.2617	0.5370	0.2896	0.4605

Table 10: Validation results for Sub-task C.

TUVD team at SemEval-2019 Task 6: Offense Target Identification

Elena Shushkevich
Social Media
Research Group
Technological University
Dublin, Ireland
e.shushkevich
@yandex.ru

John Cardiff
Social Media
Research Group
Technological University
Dublin, Ireland
john.cardiff
@it-tallaght.ie

Paolo Rosso
PRHLT Research Center
Universitat Politecnica
de Valencia, Spain
prosso@dsic.upv.es

Abstract

This article presents our approach for detecting a target of offensive messages in Twitter, including Individual, Group and Others classes. The model we have created is an ensemble of simpler models, including Logistic Regression, Naive Bayes, Support Vector Machine and the interpolation between Logistic Regression and Naive Bayes with 0.25 coefficient of interpolation. The model allows us to achieve 0.547 macro F1-score.

1 Introduction

Nowadays aggressive language on social media occurs more and more often. Categories of hate speech can be very diverse and can deal with a wide range of issues such as misogyny, sexual orientation, religion and immigration. Such types of speech can be found in posts in social networks, in Internet discussions, in comments on various articles and in responses to posts of famous persons.

This problem is receiving increasing amounts of attention and researchers are making attempts to build systems capable of recognizing such kinds of aggressive speech, offenses and insults in social networks.

This article presented our approach to hate speech detection, which we used for the challenge SemEval-2019 Task 6: OffensEval - Identifying and Categorizing Offensive Language in Social Media (Zampieri et al., 2019a ; Zampieri et al., 2019b).

The task consisted of three sub-tasks and proposed to investigate the data extracted from Twitter for creating a classification system.

Sub-task A had the aim to identify offensive language and there were 860 unmarked English tweets for testing. The post had to be non offensive if it did not contain any offense or profanity.

The main goal of the Sub-task B was to categorize offensive posts from Sub-task A (there were 240 English tweets for testing) to different offensive types:

- Targeted Insults and Threats in cases when a post insults or treats to an individual, a group or an organization;

- Untargeted in cases where a post has a non-targeting profanity and swearing.

Sub-task C focused on offense target identification. There were 213 English tweets which were marked as offensive in Sub-task A and Targeted Insult and Threats in Sub-task B for testing. The classification was for three different groups:

- Individual, when the target of the offensive post was a person;

- Group, when the target of the offensive message was a group of people considered as a unit;

- Other, when the target of the offensive tweet did not belong to any of the previous categories (e.g., a situation, an event, or another issue).

There are two datasets in English and in Spanish languages for analysis, and our team worked with English only. The training dataset included 13200 tweets, 4400 of them were offensive ones, 3876 messages were labeled as 'Target Insult and Threats' and 524 ones as 'Untargeted'. We focused our efforts on Sub-task C only, and the training dataset for it consisted of 2407 'individual' offensive posts, 1074 'group' ones and 395 tweets marked as 'other'.

The paper is organized as follows. Some relevant related works in the area are described in Section 2. Section 3 presents the preprocessing we applied for the dataset and the methodology we used for the model creating. In Section 4 the results are described and analyzed. In Section 5 we summarize our work and plan some steps for the future researches.

2 Related Work

Today there are a lot of promising works in the area of the hate speech recognition As was shown in (Fasoli et al., 2015), offensive language can be very diverse and the level of the messages offensiveness can depend on the context and the relationships between users who take part in the conversation.

For example, insults delivered in a sexual context are less offensive in cases where there is a conversation between partners. Some slurs have more offensive meaning in cases of conversations between a superior and a subordinate compared with conversations between friends and some groups of slurs are more acceptable then others.

Expanding the point that offensive speech is heterogeneous, the work (Clarke and Grieve, 2017) presented results which showed that there is a difference between racist and sexist posts: the sexist messages were more interactive (more personal) and more attitudinal (with authors opinion) than racist ones. From this article we can make a conclusion that the most popular linguistic feature in offensive language are question marks and question DO (when a sentence stars with the word do).

The work (Saleem et al., 2017) demonstrated that messages may not include slurs, but still be offensive. The authors took as training dataset messages from potentially vulnerable communities (like groups of Afro-American and plus-size users) and messages from haters of these communities (not included slurs only) and showed that the system of hate speech recognition based on traditional methods like Logistic Regression could indicate insult meaning on the posts without slurs. .

In addition, this work shows that it is possible to test dataset from one source using training set from another one. Authors checked this fact, used the training dataset from one source and the testing dataset from the another source. The results were quite good and it is allow us to say that it could be useful to add to our training dataset some comments from another social media to make predictions better.

At the Automated Misogyny Detection (AMI) Shared Tasks IBEREVAL-2018 (Fersini et al., 2018) and EVALITA-2018 (Caselli et al., 2018), some interesting approaches for offensive language detecting were presented. The main goal in these challenges was to detect misogynistic tweets and to classify tweets for different groups depending on a misogyny type (stereotypes and objectification, dominance, derailing, sexual harassment and threats of violence and discredit) and an insults target (the idea of this type of classification was to recognize misogynous tweets which offend a specific person and tweets which insult a group of people).

In (Pamungkas et al., 2018) it was shown that the results of the model based on Support Vector Machine were quite good and in the research (Frenda et al., 2018) the ensemble of models allow to achieve a high level of accuracy. In work (Shushkevich and Cardiff, 2018) it was presented the ensemble of Logistic Regression. Support Vector Machine and Naive Bayes model which shown quite good results.

It is necessary to add that models based on neural networks show good results of offensive language recognition, as it was shown in (Badjatiya et al., 2017), where the authors created the model based on Long Short-Term Networks (LSTMs) which use internal memory for capture the long range dependencies in sentences and it could be important for the hate speech detection. This approach allowed them to achieve very high results in sexist and racist tweets detection in comparison with classifiers such as Logistic Regression, Random Forest, SVMs and Gradient

Boosted Decision Trees (GBDTs).

3 Methodology and Data

As the preprocessing step we:

- converted the words to the lower case;

- used TF-IDF (Term Frequency - Inverse Document Frequency) for the vectorization;

- marked emojis with the word 'EMOJI';

- labeled some combinations of symbols like '!!! ' and '??? ', because they look like emotional expressions and could be presented as emojis too, and replaced them with the word 'EMOJI'

Our model presents an ensemble of some classic machine learning models:

- The model based on Logistic Regression (LR) (Wright, 1995; Genkin et al., 2007), this type of classifiers apply an exponential function to a lineal combination of objects extracted from the data.

- The model based on Naive Bayes (NB), whose advantages are an absence of big training dataset and speed calculations requirement (Hi and Li, 2007).

- The model presented an interpolation between LR and NB with 0.25 coefficient of interpolation as a form of regulation: trust NB unless the LR. This type of interpolation was shown in (Wang and Manning, 2012) where NB was combined with Support Vector Machine, but in our case the combination LR+NB worked better.

- The model based on Support Vector Machine (SVM), the effectiveness of which in the work with texts was described in (Joachims, 2002).

We blended all above-described models into one which indicated the belonging of a tweet to the classes according to the rule: we summarized probabilities of belonging to all three classed which all four models presented and divided this number by 4. A post was assigned a class with the highest average probability.

4 Results

The predicted results of F1-macro for the all 5 models are presented in Table 1.

As it shown the Blended model achieves the highest score (0.68), so we could conclude that our hypothesis was correct and an ensemble of models presented the best results for the task of offense target identification.

Also, the model which combine Logistic Regression and Naive Bayes achieves good result (0.65), and the worst model for this type of classification was Logistic Regression one.

The results of the challenge are presented in Table 2. Overall Accuracy for the test set was equal to 0.6478 and Macro-F1 was 0.547.

As we can see, the macro F1-score is less when predicted with the training dataset macro F1-score by 0.133, and this difference could be connected with the small number of tweets for training. Also it should be noted, that the results of classification have a strict correlation with the number of testing examples: the IND classifier works better then GRP one and much better then OTH classifier, because in the testing dataset there were more data about individual target of offenses then about group and other targets.

5 Conclusion

To sum up, we created an ensemble of models, which allow as to achieve quite good results being placed 25th out of a total of 65 participants. We showed that the idea of blending simple models based on Logistic Regression, Naive Bayes and Support Vector Machine gives a perspective in the area of hate speech recognition in the identification of the target of offensive messages.

As the next steps in our research, we are planning to expand the preprocessing step and use some dictionaries and lists of offensive language, which could help us to achieve better results. We also intend to additional data for the training datasets.

It is interesting to add, that in these datasets all links were replaced with URL and all usernames

Model	F1 (macro)
Logistic Regression	0.50
Naive Bayes	0.63
LR+NB	0.65
Support Vector Machine	0.60
Blended Model	**0.68**

Table 1: Results for each model with training dataset for the Subtask C

Type of classification	F1-score
GRP	0.6047
IND	0.7615
OTH	0.2759
avg/totall	**0.6243**

Table 2: Results of the classification with testing dataset for the Subtask C

in tweets were replaced with USER. It could be useful to investigate, for example, links, which were mentioned in offensive messages. It could be possible to expand our dataset in cases when link was a respond for another offensive post or we could lable tweets which have links for a blocked content.

In this challenge we faced the problem of the an insufficient quantity of tweets to make our classifier work better: for example, for the class Other there were only 395 post for training. We believe that an increase in the volume of data could make our modeling more effective, and external data sources could be helpful. Also, we intend to experiment with the use of LSTMs.

References

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion pp. 759760.*

Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso. 2018. Overview of the Evalita 2018 Task on Automatic Misogyny Identification (AMI)). In *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18) CEUR Workshop Proceedings. CEUR.org*, volume 2263, Turin, Italy.

Isobelle Clarke and Jack Grieve. 2017. Dimensions of abusive language on twitter. In *American Psychological Association*, Washington, DC.

Fabio Fasoli, Andrea Carnaghi, and Maria Paola Paladino. 2015. Social acceptability of sexist derogatory and sexist objectifying slurs across contexts. In *Language Sciences, 52.*

Elisabetta Fersini, Maria Anzovino, and Paolo Rosso. 2018. Overview of the Task on Automatic Misogyny Identification at IberEval). In *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018), co-located with 34th Conference of the Spanish Society for Natural Language Processing (SEPLN 2018) CEUR Workshop Proceedings. CEUR-WS.org*, volume 2150, Seville, Spain.

Simona Frenda, Bilal Ghanem, and Manuel Montes-y Gomez. 2018. Exploration of Misogyny in Spanish and English tweets. In *CEUR Workshop Proceedings. CEUR-WS.org*, volume 2150, Seville, Spain.

Alexander Genkin, David Lewis, and David Madigan. 2007. Large-scale bayesian logistic regression for text categorization. In *Technometrics, 49(3):291304.*

Zhang Hi and Di Li. 2007. Naive bayes text classifier. granular computing. In *IEEE International Conference on, pages 708708. IEEE.*

Thoarsten Joachims. 2002. Learning to classify text using support vector machines: Methods, theory and algorithms. In *Kluwer Academic Publishers.*

Endang Wahyu Pamungkas, Alessandra Teresa Cignarella, Valerio Basile, and Viviana Patti. 2018. P14-ExLab@UniTo for AMI at IberEval2018: Exploiting Lexical Knowledge for Detecting Misogyny in English and Spanish Tweets. In *CEUR Workshop Proceedings. CEUR-WS.org*, volume 2150, Seville, Spain.

Haji Mohammad Saleem, P. Dillon Kelly, Susan Benesch, and Derek Ruths. 2017. A web of hate: Tackling hateful speech in online social spaces. In *CoR abs/1709.10159.*

Elena Shushkevich and John Cardiff. 2018. Classifying Misogynistic Tweets Using a Blended Model: The AMI Shared Task in IBEREVAL 2018. In *CEUR Workshop Proceedings. CEUR-WS.org*, volume 2150, Seville, Spain.

Sida Wang and Christopher D. Manning. 2012. Baselines and bigrams: simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers, ACL, vol. 2, pp. 9094*.

Robert Wright. 1995. Logistic regression. In *Proceedings of the 26th International Conference on World Wide Web Companion pp. 759760.* L.C.Grimm.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

UBC-NLP at SemEval-2019 Task 6: Ensemble Learning of Offensive Content With Enhanced Training Data

Arun Rajendran **Chiyu Zhang** **Muhammad Abdul-Mageed**
Natural Language Processing Lab
University of British Columbia
muhammad.mageed@ubc.ca

Abstract

We examine learning offensive content on Twitter with limited, imbalanced data. For the purpose, we investigate the utility of using various data enhancement methods with a host of classical ensemble classifiers. Among the 75 participating teams in SemEval-2019 sub-task B, our system ranks 6th (with 0.706 macro F_1-score). For sub-task C, among the 65 participating teams, our system ranks 9th (with 0.587 macro F_1-score).

1 Introduction

With the proliferation of social media, millions of people currently express their opinions freely online. Unfortunately, this is not without costs as some users fail to maintain the thin line between freedom of expression and hate speech, defamation, ad hominem attacks, etc. Manually detecting these types of negative content is not feasible, due to the sheer volume of online communication. In addition, individuals tasked with inspecting such types of content may suffer from depression and burnout. For these reasons, it is desirable to build machine learning systems that can flag offensive online content.

Several works have investigated detecting undesirable (Alshehri et al., 2018) and offensive language online using traditional machine learning methods. For example, Xiang et al. (2012) employ statistical topic modelling and feature engineering to detect offensive tweets. Similarly, Davidson et al. (2017) train multiple classifiers (e.g., logistic regression, decision trees, and support vector machines) to detect hate speech from general offensive tweets. More recently, deep artificial neural networks (i.e., deep learning) has been used for several text classification tasks, including detecting offensive and hateful language. For example, Pitsilis et al. (2018) use recurrent neural networks (RNN) to detect offensive language in tweets. Mathur et al. (2018) use transfer learning with convolutional neural networks (CNN) for offensive tweet classification on Twitter data.

Most of these works, however, either assume relatively balanced data (traditional classifiers) and/or large amounts of labeled data (deep learning). In scenarios where only highly imbalanced data are available, it becomes challenging to learn good generalizations. In these cases, it is useful to employ methods with good predictive power for especially minority classes. For example, methods capable of enhancing training data (e.g., by augmenting minority categories) are desirable in such scenarios. In the literature, some works have been undertaken to address issues of data imbalance in language tasks. For example, Mountassir et al. (2012) propose different undersampling techniques that yield better performance than common random undersampling on sentiment analysis. Along similar lines Gopalakrishnan and Ramaswamy (2014) propose a modified ensemble based bagging algorithm and sampling techniques that improve sentiment analysis. Further, Li et al. (2018) present a novel oversampling technique that generates synthetic texts from word spaces.

In addition to data enhancement, combining various classifiers in an ensemble fashion can be useful since different classifiers have different learning biases. Past research has shown the effectiveness of ensembling classifiers for text classification (Xia et al., 2011; Onan et al., 2016). Omar et al. (2013), for example, study the performance of ensemble models for sentiment analysis of Arabic reviews. Da Silva et al. (2014) exploit ensembles to boost the accuracy on twitter sentiment analysis. Wang and Yao (2009) demonstrate the utility of combining sampling techniques with ensemble models for solving the data imbalance problem.

In this paper, we describe our submissions to SemEval-2019 task 6 (OffenseEval) (Zampieri et al., 2019b). We focus on sub-tasks B and C. The Offensive Language Identification Dataset (Zampieri et al., 2019a), the data released by the organizers for each of these sub-tasks, is extremely imbalanced (see Section 2). We propose effective methods for developing models exploiting the data. Our main contributions are: (1) we experiment with a number of simple data augmentation methods to alleviate class imbalance, and (2) we apply a number of classical machine learning methods in the context of ensembling to develop highly successful models for each of the competition sub-tasks. Our work shows the utility of the proposed methods for detecting offensive language in absence of budget for performing feature engineering and/or small, imbalanced data.

The rest of the paper is organized as follows: We describe the datasets in Section 2. We introduce our methods in Section 3. Next, we detail our models for each sub-task (Sections 4 and 5). We then offer an analysis of the performance of our models in Section 6, and conclude in Section 7.

2 Data

As mentioned, *OffenseEval* is SemEval-2019 task 6. The task is focused on identifying and categorizing offensive language in social media and involves three different sub-tasks. These are:

- **Sub-task A** is offensive language identification, e.g. classifying the given tweets into *offensive* or *non-offensive*. In our work, we only focus on sub-tasks B and C and so we do not cover sub-task A further.

- **Sub-task B** is automatic categorization of offensive content types, which involves categorizing tweets into *targeted* and *untargeted* threats. The dataset for this sub-task consists of 4,400 tweets (3,876 *targeted* and 524 *untargeted*). Table 1 provides one examples of each of these two classes.

- **Sub-task C** is offense target identification and includes the 3 classes of targets. These classes are in the set {*individual, group, others*}. The dataset for this sub-task consists of 3,876 tweets (2,407 *individual*, 1,074 *group*, and 395 *other*). We similarly provide one example for each of these classes in Table 1.

We use 80% of the tweets as our training set and the remaining 20% as our validation set for both sub-tasks B and C. We also report our best models on the competition test set, as returned to us by organizers. Table 2 provides statistics of our data for sub-tasks B and C.

3 Methods

3.1 Pre-Processing

We utilize a simple data pre-processing pipeline involving lower-casing all text, filtering out URLs, usernames, punctuation, irrelevant characters and emojis, and splitting text into word-level tokens.

3.2 Data Intelligence Methods

We employ multiple machine learning methods and combine them with different sampling and data generation techniques to enhance our training set. From a data sampling perspective, the most common approaches to deal with imbalanced data is random oversampling and random undersampling (Lohr, 2009; Chawla, 2009). Learning with these basic techniques is usually effective due to possibly reducing model bias towards the majority class. We employ a number of data sampling techniques, as described next.

Random oversampling technique randomly duplicates the minority samples to obtain a balanced dataset. Despite the naive approach, this method is reported to perform well (as compared to other sophisticated oversampling methods) in the literature. One major drawback of this method is that it does not add any new data to the training set (since it only duplicates minority-class training data) (Liu et al., 2007).

Synthetic minority over-sampling (SMOTE) is a sophisticated oversampling technique where synthetic samples are generated and added to the minority class. For each data point, one of k minority class neighbours is randomly selected and the new synthetic point is a random point on the line joining the actual data point and this randomly selected neighbour. This method has been shown to be effective compared to some other oversampling methods (Chawla et al., 2002; Batista et al., 2004).

Random undersampling removes instances from the majority class in a random manner to ob-

Task	Label	Example
Sub-task B	targeted	*Liberals are all Kookoo !!!*
	untargeted	*Dont believe the hype.*
Sub-task C	individual	*Good move...he is the big loser*
	group	*The Liberals are mentally unstable!!*
	other	*Google go to hell!*

Table 1: Examples of each class in sub-tasks B and C

Task	Label	Train	Dev	Total
Sub-task B	targeted	3,101	775	3,876
	untargeted	419	105	524
Sub-task C	individual	1,925	482	2,407
	group	859	215	1,074
	other	316	79	395

Table 2: Distribution of classes over our data splits

tain a balanced dataset. One possible disadvantage of this method is that it might remove valuable information from training data since, due to its randomness, it does not pay consideration to the data points removed (Liu et al., 2007).

kNN-based undersampling is an alternative undersampling technique (Mani and Zhang, 2003) which uses distance between points within a class. We use three different methods to select near-miss samples, as described in Mani and Zhang (2003). **NearMiss-1** selects majority class samples whose average distance to three closest minority class samples is smallest. In **NearMiss-2**, the samples of the majority class are selected such that their average distances to three farthest samples of minority class are smallest. **NearMiss-3** picks a given number of the closest majority class samples from each minority class sample, which guarantees every minority class sample is surrounded by some majority class points. Mani and Zhang (2003) choose the majority class samples whose average distances to the three closest minority class samples are farthest.

Synthetic Data Generation. We experiment with adding information to the minority class by generating synthetic samples employing a word2vec-aided paraphrasing technique. Initially, we train a word2vec model on the entire training data and use this word2vec model to generate samples for the minority class by randomly replacing words in tweets (with a probability of 0.9). We randomly pick one word from k word2vec most similar words. We fix k=5 words and probability value as 0.9, but these are hyperparameters that can be optimized. In this way, we generate a balanced dataset in an attempt to overcome the problem of imbalance. In this technique, we draw inspiration from (Li et al., 2018) where authors propose a sentiment lexicon generation method using a label propagation algorithm and utilize the generated lexicons to obtain synthetic samples for the minority class by randomly replacing a set of words with words that have similar semantic content.

3.3 Classifiers

We apply a number of machine learning classifiers that are proven to work well for text categorization. Namely, we use logistic regression, support vector machines (SVM) and Naive Bayes. We also experiment with boosting algorithms such as random forest, AdaBoost, bagging classifier, XGBoost, and gradient boosting classifier. We deploy ensembles of our best performing models in two ways: (1) ensembles based on majority rule classifiers that use predicted class labels for majority rule voting and (2) soft voting classifiers that predict the class label based on the argmax of the sums of the predicted probabilities of various classifiers.

4 Sub-Task B Models

For sub-task B, we have one minority class, so we generate samples for this minority class to obtain a new, balanced dataset. We use this balanced data as well as the the imbalanced (ORG) dataset for our first iteration of experiments. The goal of iteration is to identify the best (1) input n-gram settings (explained next), (2) classifier (from our classifiers listed in Section 3.3, and (3) sampling techniques (listed in 3.2). For **n-gram settings**, we use a combination of bag of words and TF-IDF to extract features from the tweets and run with unigrams and all different combinations of unigram, bigrams, trigrams, and four grams. We run on all combinations across all the three variables above (n-grams, classifiers, and sampling methods) on both the imbalanced (ORG) and balanced datasets. Since our datasets are small, this iteration of experiments is not very costly. We acquire best results on the balanced dataset, identifying the combination of unigrams and bigrams as our best n-gram settings, XGBoost as the best classifier, and SMOTE as the best sampling technique. We provide these best results in Table 3 in Macro-F1 score. We use two baselines. Baseline 1 is the majority class in training data (i.e., *targeted* offense class, 0.46827 Macro F_1-score). The second baseline is the best model with no data sampling, a logistic regression model. The best model, XGBoost with SMOTE sampling, acquires an F_1-score of 0.61248. This is a sizeable gain over the baselines. We now describe how we leverage ensembles to improve over this XGBoost model.

Sampling Type	**Sampling Technique**	**Macro F1**
NA	**Baseline 1**	0.46827
	Baseline 2	0.5547
Oversampling	**Random Oversampling**	0.56705
	SMOTE	**0.61248**
Undersampling	**Random Undersampling**	0.49739
	Near Miss-1	0.32533
	Near Miss-2	0.4376
	Near Miss-3	0.46158

Table 3: **Sub-Task B:** XGBoost performance with sampling methods. Baseline 1 is our majority class in training data. Baseline 2 is a logistic regression model with no data sampling.

4.1 Ensembles for Sub-Task B

Our best performance with the XGBoost model in the previous section was acquired with SMOTE oversampling. However, we note that oversampling in general performed better than other sampling methods. For this reason, we experiment with a number of ensemble methods across our two oversampling techniques (SMOTE and random oversampling [ROS]). We provide our best results from this iteration of experiments (for both the dev and the competition test set) in Table 4. In addition to the same XGBoost model reported earlier (in Table 3, reproduced in Table 4), we identify and report our two best models: (1) **Model A**: An ensemble with soft voting over XGBoost, AdaBoost, and logistic regression with random oversampling (ROS) and (2) **Model B**: The average of our XGBoost model (with SMOTE) and the best model with synthetic oversampling (which is a Naive Bayes classifier). We submitted the three models in Table 4 to the competition. Although Model B performs best on the dev set, it was model A that performed highest on the competition test set. This suggests that the dev and test sets are different in some aspects. Importantly, even though the three models in Table 4 perform comparably on dev, only the ensemble models (Model A and Model B) seem to generalize better on the test set. This further demonstrates the utility of ensembles on the task.

5 Sub-Task C Models

Sub-Task C is 3-way classification, with 2 minority classes. Again, we run all our classifiers with unigram and bigram combinations across all sampling methods (including no sampling) on this imbalanced dataset. In addition, we use 4 different configurations to generate samples for each of the two minority classes to obtain 4 balanced datasets. `C1` is created with random oversampling of the two minority classes; `C2` is created with synthetic oversampling of the two minority classes; `C3` is created with random oversampling of minority class *group* (GRP) and synthetic oversampling of minority class *other* (OTH); and `C4` is random oversampling of minority class OTH and synthetic oversampling of minority class GRP.

We report our best results in Table 5, with two baselines: Baseline 1 is the majority class in training data and Baseline 2 is our best model without sampling (a logistic regression classifier). Our best model on C2 is a logistic regression classifier, whereas our best models on C1, C3, and C4 are acquired with the same soft voting ensemble in Table 4 (an ensemble of logistic regression, AdaBoost, and XGBoost).

Our next step is to investigate whether we can further improve performance by averaging classification probabilities of models described in Table 5. The result of this iteration is shown in Table 6. Models in Table 6 are the 3 models we submitted to the SemEval-2019 competition, which are as

Dataset	Models	Targeted			Untargeted			
		Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Macro F1 score**
DEV	**XGBoost (SMOTE)**	0.90158	**0.95742**	**0.92866**	**0.42105**	0.22857	0.2963	0.61248
	Model A	0.90945	0.89419	0.90176	0.30508	0.34286	**0.32287**	0.61231
	Model B	**0.94065**	0.90447	0.9222	0.26667	**0.37838**	0.31285	**0.61753**
TEST	**XGBoost (SMOTE)**	0.9004	**0.9765**	0.9369	0.4444	0.1481	0.2222	0.57958
	Model A	**0.9378**	0.9202	0.9289	0.4516	**0.5185**	**0.4828**	**0.70583**
	Model B	0.9079	0.9718	**0.9388**	**0.5000**	0.2222	0.3077	0.62323

Table 4: **Sub-Task B:** Best ensemble model results. We reproduce XGBoost results from Table 3 for comparison.

Sampling	Type	Best Model	Macro F1
NA	NA	Baseline-1	0.21300
	NA	Baseline-2	0.51580
Sampling	**C1**	**Model 1**	**0.56822**
	C2	Log Reg	0.54319
	C3	Model 1	0.54665
	C4	Model 1	0.56216

Table 5: **Sub-Task C:** Best results with various sampling methods.

follows: **Model 1**: our best model with C1; **Model 2**: a prediction based on the average of classification probabilities of the best classifiers on C1, C2, and C4; **Model 3**: the prediction acquired from the average of tag probabilities of the best classifiers on C1 and C4. Table 6 shows that performance of all the models on the dev set is very comparable, with model 3 performing slightly better than the two other models. Similarly, results of the three models are not very different on the competition test set.

6 Model Analysis

In order to further understand the results on the test set, we investigate the predictions made by our models across the two sub-tasks. For the purpose, we provide simple visualizations of the confusion matrices of predictions acquired by our best models as released by organizers.

Sub-Task B. Figure 1 shows that our model has higher precision for the targeted threats, which is also clear from Table 4 presented earlier. Figure 1 also shows that our model has slightly higher false negatives as compared to false positives. In other words, the chances of our model mislabeling a *targeted* tweet as *untargeted* is slightly higher as compared to predicting an *untargeted* tweet as *targeted*.

Sub-Task C We visualize model errors in Figure 2. Figure 2 shows that our model has

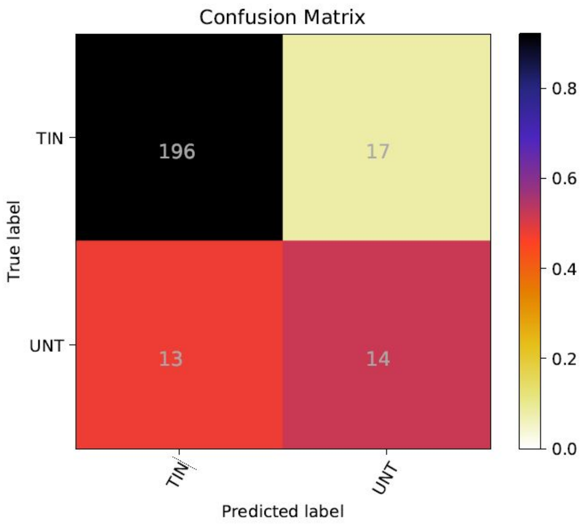


Figure 1: Confusion matrix of soft voting ensemble model (Model A in Table 4) for Sub-Task B.

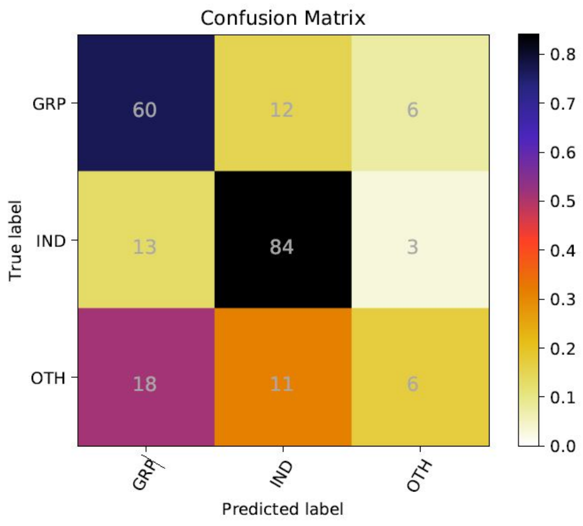


Figure 2: Confusion matrix of soft voting ensemble model (Model 1 in Table 6) for Sub-Task C.

Dataset	Models	GRP	IND	OTH	Macro F1 score
DEV	Model 1	0.60538	**0.82476**	0.27451	0.56822
	Model 2	**0.61504**	0.8204	0.28931	0.57492
	Model 3	0.61207	0.8203	**0.29577**	**0.57605**
TEST	Model 1	**0.7101**	0.8116	0.2400	**0.58722**
	Model 2	0.686	0.8098	0.2041	0.56663
	Model 3	0.6946	**0.819**	**0.2449**	0.58619

Table 6: **Sub-Task C:** Results of our 3 final submitted models

higher precision for the *group* (GRP) and *individual* (IND) categories, but only higher recall for the *other* (OTH) class. Again, this means that the chances of our model predicting a GRP tweet or IND tweet as OTH is much higher as compared to OTH tweet being predicted as IND or GRP. In other words, the model is biased towards predicting one of the two categories GRP and IND

7 Conclusion

In this paper, we described our contributions to OffenseEval, the 6th shared task of SemEval-2019 . We explored the effectiveness of different sampling techniques and ensembling methods combined with different classical and boosting machine learning algorithms. We find simple data enhancement approaches (i.e., sampling techniques) to work well, especially when coupled with the right ensemble methods. In general, ensemble models decrease errors by leveraging the different strengths of the various underlying models and hence are useful in absence of balanced data.

8 Acknowledgement

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Social Sciences Research Council of Canada (SSHRC). The research was partially enabled by WestGrid (www.westgrid.ca) and Compute Canada (www.computecanada.ca).

References

Ali Alshehri, AlMoetazbillah Nagoudi, Alhuzali Hassan, and Muhammad Abdul-Mageed. 2018. Think before your click: Data and models for adult content in arabic twitter. *The 2nd Text Analytics for Cybersecurity and Online Safety (TA-COS-2018), LREC.*

Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. 2004. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20–29.

Nitesh V Chawla. 2009. Data mining for imbalanced datasets: An overview. In *Data mining and knowledge discovery handbook*, pages 875–886. Springer.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Nadia FF Da Silva, Eduardo R Hruschka, and Estevam R Hruschka Jr. 2014. Tweet sentiment analysis with classifier ensembles. *Decision Support Systems*, 66:170–179.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Eleventh International AAAI Conference on Web and Social Media.*

Vinodhini Gopalakrishnan and Chandrasekaran Ramaswamy. 2014. Sentiment learning from imbalanced dataset: an ensemble based method. *Int. J. Artif. Intell*, 12(2):75–87.

Yijing Li, Haixiang Guo, Qingpeng Zhang, Mingyun Gu, and Jianying Yang. 2018. Imbalanced text sentiment classification using universal and domain-specific knowledge. *Knowledge-Based Systems*, 160:1–15.

Alexander Liu, Joydeep Ghosh, and Cheryl E Martin. 2007. Generative oversampling for mining imbalanced datasets. In *DMIN*, pages 66–72.

Sharon L Lohr. 2009. *Sampling: design and analysis.* Nelson Education.

Inderjeet Mani and I Zhang. 2003. knn approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of workshop on learning from imbalanced datasets*, volume 126.

Puneet Mathur, Rajiv Shah, Ramit Sawhney, and Debanjan Mahata. 2018. Detecting offensive tweets in hindi-english code-switched language. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, pages 18–26.

Asmaa Mountassir, Houda Benbrahim, and Ilham Berrada. 2012. Addressing the problem of unbalanced data sets in sentiment analysis. In *KDIR*, pages 306–311.

Nazlia Omar, Mohammed Albared, Adel Qasem Al-Shabi, and Tareq Al-Moslmi. 2013. Ensemble of classification algorithms for subjectivity and sentiment analysis of arabic customers' reviews. *International Journal of Advancements in Computing Technology*, 5(14):77.

Aytuğ Onan, Serdar Korukoğlu, and Hasan Bulut. 2016. Ensemble of keyword extraction methods and classifiers in text classification. *Expert Systems with Applications*, 57:232–247.

Georgios K Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*.

Shuo Wang and Xin Yao. 2009. Diversity analysis on imbalanced data sets by using ensemble models. In *2009 IEEE Symposium on Computational Intelligence and Data Mining*, pages 324–331. IEEE.

Rui Xia, Chengqing Zong, and Shoushan Li. 2011. Ensemble of feature sets and classification algorithms for sentiment classification. *Information Sciences*, 181(6):1138–1152.

Guang Xiang, Bin Fan, Ling Wang, Jason Hong, and Carolyn Rose. 2012. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1980–1984. ACM.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

UHH-LT at SemEval-2019 Task 6: Supervised vs. Unsupervised Transfer Learning for Offensive Language Detection

Gregor Wiedemann **Eugen Ruppert*** **Chris Biemann**
Language Technology Group / *Base.Camp
Department of Informatics
University of Hamburg, Germany
{gwiedemann, ruppert, biemann}@informatik.uni-hamburg.de

Abstract

We present a neural network based approach of transfer learning for offensive language detection. For our system, we compare two types of knowledge transfer: supervised and unsupervised pre-training. Supervised pre-training of our bidirectional GRU-3-CNN architecture is performed as multi-task learning of parallel training of five different tasks. The selected tasks are supervised classification problems from public NLP resources with some overlap to offensive language such as sentiment detection, emoji classification, and aggressive language classification. Unsupervised transfer learning is performed with a thematic clustering of 40M unlabeled tweets via LDA. Based on this dataset, pre-training is performed by predicting the main topic of a tweet. Results indicate that unsupervised transfer from large datasets performs slightly better than supervised training on small 'near target category' datasets. In the SemEval Task, our system ranks 14 out of 103 participants.

1 Introduction

The automatic detection of hate speech, cyberbullying, abusive, aggressive or offensive language has become a vital field of research in natural language processing (NLP) during recent years. Especially in social media, the tone of conversations escalates in a disturbing way that often threatens a free, democratic and argumentative discourse of users. Concerning the tremendous amount of digital texts posted on platforms such as Twitter, Facebook or in comments sections of online newspapers, automatic approaches to offensive language detection are of high relevancy for moderation and filtering of content as well as for studying the phenomenon of offensive language use in social media at large scale.

To take account of this development, a shared task on "offensive language detection" was conducted at the SemEval 2019 workshop. This paper describes our approach to the shared task 6 (OffensEval) as organized and described in detail by Zampieri et al. (2019b). The task contains three hierarchically ordered sub-tasks: Task A requires a classification of tweets into either offensive (OFF) or not offensive (NOT), Task B subdivides all offensive tweets into either targeted insults (TIN) or generally offensive expressions not targeted to any specific entity (UNT), and Task C finally asks to assign one out of three specific target labels to all targeted insults: groups (GRP, e.g. ethnic groups), individuals (IND, e.g. a politician or a specific Twitter user), or other (OTH, e.g. the media industry). The dataset consisting of 14,100 examples (13,240 in the training set, 860 in the test set) was annotated via crowdsourcing (Zampieri et al., 2019a). Each tweet was labeled by at least two annotators who must reach an agreement of at least 66% for including the tweet in the dataset. The dataset is characterized by a high imbalance of label distributions, especially for Tasks B and C.

There are several challenges for automatic offensive language detection that render simple dictionary-based approaches unusable. First, label distribution in the dataset is highly skewed for all sub-tasks. Although offensive language is a growing problem for social media communication, it still accounts for only a small fraction of all content posted. Second, language characteristics in social media pose a severe challenge to standard NLP tools. Misspellings, slang vocabulary, emoticons and emojis, as well as ungrammatical punctuation must be taken into account for a successful solution. Third, offensive language is highly context-dependent. For instance, swear words are often used to mark overly positive emotion ("This is fucking great!!!"), and actually neutral and descriptive sentences can be conceived as derogatory if they refer to a specific individual ("@Barack-

Obama he is a Muslim").

Our approach to the OffensEval shared task is based on two main contributions: First, we introduce a BiGRU-3CNN neural network architecture in combination with pre-trained sub-word embeddings that are able to handle social media language robustly. Second, we investigate two types of knowledge transfer: supervised and unsupervised pre-training. Supervised pre-training of our neural network architecture is performed as multi-task learning of parallel training of five different NLP tasks with some overlap to offensive language detection. Unsupervised transfer learning is performed with a thematic clustering of a large dataset of unlabeled tweets via LDA. After shortly referencing related work (Section 2), we introduce both approaches in detail in Section 3 and present the results in Section 4.

2 Related Work

Two recent survey papers, Schmidt and Wiegand (2017) and Fortuna and Nunes (2018), summarize the current state of the art in offensive language detection and related tasks such as hate speech or abusive language detection. Specifically for offensive language detection, the paper by Davidson et al. (2017) introduced a publicly available dataset which was reused in (Malmasi and Zampieri, 2017, 2018; ElSherief et al., 2018; Zhang et al., 2018) as well as in our approach of supervised pre-training.

A predecessor of our transfer learning approach has already successfully been applied at GermEval 2018 (Wiegand et al., 2018), a shared task on offensive language detection in German language tweets. In our paper (Wiedemann et al., 2018), we tested different types of knowledge transfer and transfer learning strategies. We further found that latent semantic clusters of user handles in tweets (e.g. user accounts run by media companies or politicians) are a very helpful feature to predict offensive language since they provide valuable context information how to interpret otherwise ambiguous tweets. Unfortunately, this feature cannot be used for the SemEval 2019 Task 6 since user mentions have all been unified to the token '@USER' in the training data. Thus, we base our approach on the best performing transfer learning strategy from Wiedemann et al. (2018) but implement several minor improvements, which will be described in detail in the following.

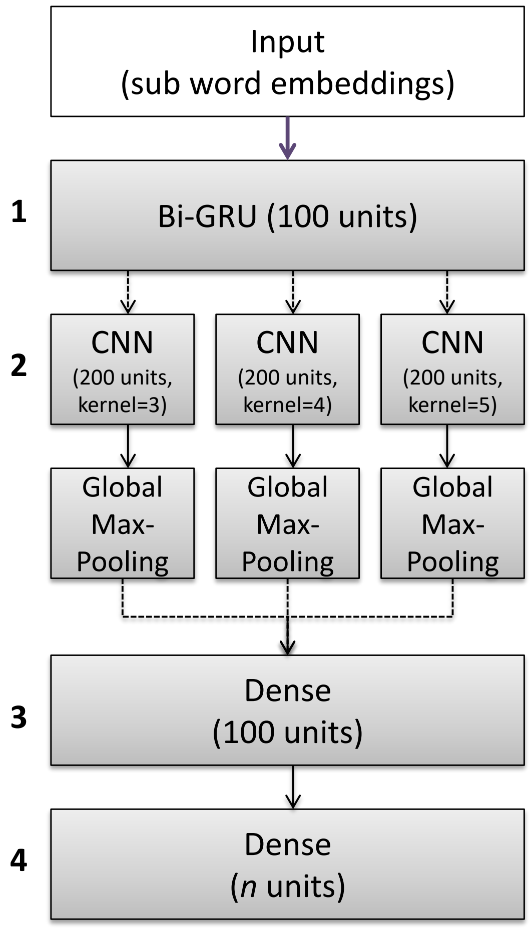


Figure 1: BiGRU-3-CNN model architecture. We use a combination of recurrent and convolutional cells for learning. As input, we rely on (sub-)word embeddings. Dashed lines indicate dropout with rate 0.5 between layers. The last dense layer contains n units for prediction of the probability of each of the n classification labels per sub-task.

3 Methodology

We utilize a neural network architecture for text classification with randomly initialized weights as a baseline, and together with two types of pre-training layer weights for transfer learning: supervised and unsupervised pre-training. Evaluation for model selection is performed via 10-fold cross-validation to determine submission candidates for the SemEval shared task.

Preprocessing: Tweets are tokenized with an extended version of the NLTK (Bird et al., 2009) tweet tokenizer. In addition to correct tokenization of emoticons and shortening of repeated character sequences (e.g. '!!!!!!') to a maximum length of three, we separate # characters as individual token from hashtags. If hashtags contain camel casing, we split them into separate tokens at each uppercase letter occurrence (e.g. '#DemocratsForPeace' is tokenized into '# democrats for peace'). Finally, all tokens are reduced to lower case. In order to account for atypical language, we use sub-word embeddings to represent the input of token sequences to our model. FastText embeddings (Bojanowski et al., 2017) are derived from character n-grams and, thus, provide meaningful word vectors even for words unseen during training, misspelled words and words specifically used in the context of social media such as emojis. We utilize a pre-trained model for the English language published by Bojanowski et al. (2017).

Model architecture: We employ a neural network architecture implemented with the Keras framework for Python[1] as shown in Fig. 1. It combines a bi-directional Gated Recurrent Unit (GRU) layer (Cho et al., 2014) with 100 units followed by three parallel convolutional layers (CNN), each with a different kernel size $k \in 3, 4, 5$, and a filter size 200. The outputs of the three CNN blocks are reduced by global max-pooling and concatenated into a single vector. This vector is then fed into a dense layer with LeakyReLU activation producing a final feature vector of length 100, which is forwarded into the prediction layer (softmax activation). For regularization, dropout is applied to the recurrent layer and to each CNN block after global max-pooling (dropout rate 0.5). For training, we use categorical cross-entropy loss and the Nesterov Adam optimization with a learning rate of 0.002. To account for imbalance in the training set, we set class weights to pay more attention to samples from the under-represented class in the loss function.

Supervised Pre-training: Instead of end-to-end text classification based on a random initialization of the parameters weights of our model, we seek to increase performance from knowledge transfer. For the supervised approach, we pre-train the model weights in a multi-task learning setup with related semantic categories. Instead of one prediction layer (see layer 4 in Fig. 1), we use m prediction layers connected to layer 3 to train m tasks in parallel. The following four publicly available datasets were compiled into one training set: offensive language tweets by (Davidson et al., 2017), flamewar comments from the Yahoo news annotated corpus (Napoles et al., 2017), sentiments of tweets from (Go et al., 2009), aggressive tweets and Facebook comments from the TRAC shared task (Kumar et al., 2018). A fifth dataset was compiled from about 30,000 randomly sampled tweets in our unsupervised background collection (see next Section) containing either a happy or an angry emoji. The merged dataset contains ca. 115k partially labeled instances for pre-training from which a sample of 5k was used as validation set. Missing labels for the combined set were filled by training a separate model for each of the m individual tasks on the respective dataset and predict a label for each instance in the other four datasets. Multi-task pre-training is performed with a batch-size of 256 for 15 epochs.

Unsupervised Pre-training: For the unsupervised approach, we utilize a large background corpus of tweets that were collected from the Twitter streaming API in 2018. Since the API provides a random fraction of all tweets (1%), language identification is performed to filter for English tweets only. From this tweet collection, we sample 20 million non-duplicate tweets containing at least two non-URL tokens as our background corpus. As a pre-training task, we first compute a Latent Dirichlet Allocation (LDA) model (Blei et al., 2003) with $K = 1,000$ topics to obtain semantic clusters of our background corpus.[2] From the topic-document distribution of the resulting LDA model, we determine the majority topic id for each tweet as a target label for prediction during pre-training our neural model. Pre-training of the neural model is performed with a batch-size of 256 for 10 epochs.

Transfer learning: Once the neural model has been pre-trained, we can apply it for learning our actual target task. For this, we need to remove the final prediction layer of the pre-trained model (i.e. Layer 4 in Fig. 1) and add a new dense layer for prediction of one of the actual label sets. To prevent the effect of "catastrophic forgetting" of pre-trained knowledge during task-specific model training, we apply a specific layer weight freezing strategy as suggested in Wiedemann et al. (2018). First, the newly added final prediction layer is trained while all other model weights re-

[1]https://keras.io

[2]For LDA, we used Mallet (http://mallet.cs.umass.edu) with Gibbs Sampling for 1,000 iterations and priors $\alpha = 10/K$ and $\beta = 0.01$.

Task	No transfer	Supervised	Unsupervised
A	76.26	**77.46**	77.36
B	58.87	**61.24**	60.57
C	56.66	54.16	**58.26**

Table 1: Model selection (cross-validation, macro-F1)

main frozen. Training is conducted for 15 epochs. After each epoch performance is tested on the validation set. The best performing model state is then used in the next step of fine-tuning the pre-trained model layers. Employing a bottom-up strategy, we unfreeze the lowest layer (1) containing the most general knowledge first, then we continue optimization with the more specific layers (2 and 3) one after the other. During fine-tuning of every single layer, all other layers remain frozen and training is performed again for 15 epochs selecting the best performing model at the end of each layer optimization. In a final round of fine-tuning, all layers are unfrozen.

Ensemble: For each sub-task A, B and C, the cross-validation results in 10 best performing models from transfer learning per configuration. For submission to the shared task, we select the model with the highest average performance across all folds. Moreover, as a simple ensemble classification, predictions of these 10 models on the test set instances are combined via majority vote.

4 Results

Model selection: To compare different types of pre-training for knowledge transfer, we use the official shared task metric macro-averaged F1. Table 1 displays the averaged results of 10-fold cross-validation for all three tasks with no transfer as baseline compared to supervised transfer from multi-task learning and pre-training on unsupervised LDA clustering. The results indicate that transfer learning is able to improve performance for offensive language detection for all tasks. With the exception of supervised transfer for task C, the relative improvements are larger the smaller the training datasets get for each of the hierarchically ordered tasks. In general, for the lower level tasks B and C, a severe performance drop can be observed compared to task A.

The comparison between unsupervised and supervised pre-training delivers a mixed result. While the performance of the supervised transfer approach slightly exceeds the unsupervised approach for task A and B, for task C, containing only very small numbers of positive examples for each class in the training dataset, the unsupervised approach clearly beats the network pre-training by supervised near-target category tasks. Supervised transfer even fails to beat the baseline of no transfer learning at all. We assume that this type of pre-training tends to over-fit the model if there is only little training data to learn from. Unsupervised pre-training on very large datasets, in contrast, better captures generic language regularities which is beneficial for arbitrary categories.

System	F1 (macro)	Accuracy
Task A		
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
Supervised	**0.7887**	**0.8372**
Unsupervised	0.7722	0.8337
Task B		
All TIN baseline	0.4702	0.8875
All UNT baseline	0.1011	0.1125
Unsupervised	**0.6608**	**0.8917**
Task C		
All GRP baseline	0.1787	0.3662
All IND baseline	0.2130	0.4695
All OTH baseline	0.0941	0.1643
Unsupervised	**0.5752**	**0.6761**

Table 2: Official test set performance results

Shared task submissions: Table 2 displays our best official results of ensemble classifications submitted to the shared tasks A, B, and C. A systematic comparison between the two compared approaches of pre-training would have required submission of two classifications per sub-task, one for supervised and one for unsupervised pre-training. Unfortunately, the official shared task website only allowed for three submissions per sub-task[3]. This policy led to the decision to submit only variations / repeated runs of the best classifier we had until the task submission deadline.

Our supervised pre-training approach ranks 14 out of 103 for sub-task A. For sub-tasks B and C, only classifiers pre-training with the unsupervised approach have been submitted. They rank 21 out of 75 for B, and 13 out of 65 for C (see

[3]Also, the official test set was not released yet, so we cannot report a systematic comparison at this point.

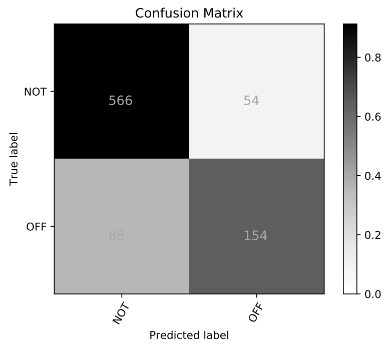


Figure 2: Sub-task A, supervised

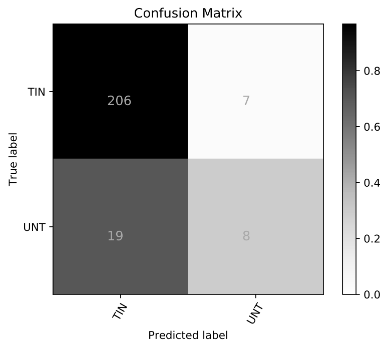


Figure 3: Sub-task B, unsupervised

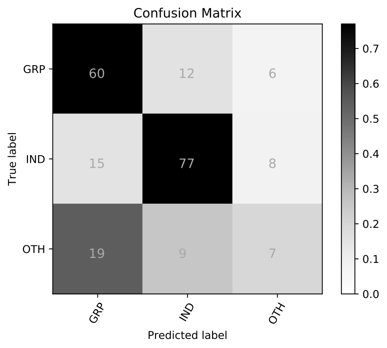


Figure 4: Sub-task C, unsupervised

Zampieri et al. (2019b) for a detailed comparison of all submissions). Fig. 2 to 4 show confusion matrices for the three best runs. The ratio between false positives and false negatives for sub-task A is fairly balanced. False positives mainly comprise hard cases where, for instance, swear words are used in a non-offensive manner. In the highly unbalanced annotations for sub-task B, more tweets were wrongly predicted as targeted insults than true yet unpredicted targeted insults. Here we observe many cases which contain offensive language and some mentioning of individuals or groups but both are not directly linked. A similar situation, where actually characteristics of two categories are contained in a tweet, can be observed for task C in which the classifier falsely predicts a group target instead of 'other'.

5 Conclusion

We systematically compared to types of knowledge transfer for offensive language detection: supervised and unsupervised pre-training of a BiGRU-3-CNN neural network architecture. The former uses a set of near-target category labeled short texts while the latter relies on a very large set of unlabeled tweets. On average, our system performs among the top 20% of all submissions of the OffensEval 2019 shared task. From our experiments, we can draw the following three main conclusions:

- Supervised pre-training with annotated near-target category data is beneficial if the target training data is fairly large.
- Unsupervised pre-training with unlabeled data from LDA clustering processes improves learning for arbitrary tasks even for fairly small target training datasets.
- For unsupervised pre-training, the benefit of transfer learning compared to the baseline without transfer is larger the smaller the target training dataset gets.

In future work, we plan to further investigate the differences between the two types of transfer learning by systematically investigating the influence of different near-target category datasets, and unsupervised topic clustering methods other than LDA for pre-training deep neural network architectures.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. ACL.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media (ICWSM)*, pages 512–515, Montreal, Canada. AAAI.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85:1–85:30.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12):1–6.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, pages 1–11, Santa Fe, NM, USA.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of the 2017 International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 467–472, Varna, Bulgaria. ACL.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Courtney Napoles, Joel Tetreault, Enrica Rosata, Brian Provenzale, and Aasish Pappu. 2017. Finding Good Conversations Online: The Yahoo News Annotated Comments Corpus. In *Proceedings of The 11th Linguistic Annotation Workshop*, pages 13–23, Valencia, Spain. ACL.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media.*, pages 1–10, Valencia, Spain. ACL.

Gregor Wiedemann, Eugen Ruppert, Raghav Jindal, and Chris Biemann. 2018. Transfer Learning from LDA to BiLSTM-CNN for Offensive Language Detection in Twitter. In *Proceedings of GermEval Task 2018, 14th Conference on Natural Language Processing (KONVENS)*, pages 85–94, Vienna, Austria.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval Task 2018, 14th Conference on Natural Language Processing (KONVENS)*, pages 1–10, Vienna, Austria.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Minneapolis, MN, USA.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval)*, Minneapolis, MN, USA. ACL.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *The Semantic Web (ESWC 2018)*, pages 745–760, Iraklio, Greece. Springer.

UM-IU@LING at SemEval-2019 Task 6: Identifying Offensive Tweets Using BERT and SVMs

Jian Zhu
Department of Linguistics
University of Michigan
Ann Arbor, MI, USA
lingjzhu@umich.edu

Zuoyu Tian
Department of Linguistics
Indiana University
Bloomington, IN, USA
zuoytian@iu.edu

Sandra Kübler
Department of Linguistics
Indiana University
Bloomington, IN, USA
skuebler@indiana.edu

Abstract

This paper describes the UM-IU@LING's system for the SemEval 2019 Task 6: OffensEval. We take a mixed approach to identify and categorize hate speech in social media. In subtask A, we fine-tuned a BERT based classifier to detect abusive content in tweets, achieving a macro F_1 score of 0.8136 on the test data, thus reaching the 3rd rank out of 103 submissions. In subtasks B and C, we used a linear SVM with selected character n-gram features. For subtask C, our system could identify the target of abuse with a macro F_1 score of 0.5243, ranking it 27th out of 65 submissions.

1 Introduction

With the increased influence of social media on modern society, large amounts of user-generated content emerge on the internet. Besides the exchange of ideas, we also see an exponential increase of aggressive and potentially harmful content, for example, hate speech. If we consider the amount of user-generated data, it is impractical to manually identify the malicious speech. Thus we need to develop methods to detect offensive speech automatically through computational models. However, this task is challenging because natural language is fraught with ambiguities, and language in social media is extremely noisy. Here we present our method to automatically identifying offensive content in tweets.

We primarily focus on detecting whether a tweet contains offensive content or not (subtask A), and then determining the target of the offensive content (subtask C). For subtask A, we use pre-trained word embeddings by fine-tuning the BERT model (Devlin et al., 2018) for detecting offensive tweets. For subtasks B and C, BERT did not perform well, either because of limited training data or because we did not find the appropriate hyperparameters. Thus we use an SVM classifier with character n-grams as features. We accidentally flipped the predicted labels in our submission to subtask B, which is why we do not report results of subtask B here. Among all teams participating in OffensEval, our models ranks 3rd out of 103 on subtask A and 27th out of 65 on subtask C. (see Zampieri et al., 2019b).

2 Related Work

Detecting offensive language online is becoming more and more important (Schmidt and Wiegand, 2017; Founta et al., 2018; Malmasi and Zampieri, 2018). To build an effective classifier, one of the major problems is to find the appropriate features. Normally, two types of features are utilized: surface features like n-grams and word representations trained by neural network. Most offensive language classifiers are trained on different types of surface features with approaches like SVM (Malmasi and Zampieri, 2018; Arroyo-Fernández et al., 2018), Random Forest (Burnap and Williams, 2015), and Logistic Regression (Davidson et al., 2017). Recently, word embeddings trained in neural networks have been shown to achieve good performance in offensive language identification tasks (Badjatiya et al., 2017). Benchmarks of the first shared task on aggression identification (Kumar et al., 2018) show that half of the top 15 systems are trained on neural networks.

Using pre-trained word embeddings for feature extraction has been shown to be highly effective in multiple NLP tasks. Traditional word embeddings are extracted from shallow neural networks trained on a large swathes of texts required to learn the contextual representations of words. Examples include skip-grams (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). However, these embeddings are learned from an aggregation of all possible word contexts, which may gloss over semantic nuances in representations.

Recent models like ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018) significantly advanced the state-of-the-art in language modeling by learning context-sensitive representations of words. ELMo goes beyond word embeddings by learning representations that are functions of the entire input sentence (Peters et al., 2018). However, ELMo is still considered shallow with two bidirectional LSTM layers, and more recent transformer based language models such as the OpenAI Generative Pre-trained Transformer (GPT) (Radford et al., 2018) and Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) have been extended to a depth of up to twelve layers. The OpenAI GPT is still a unidirectional language model while BERT is trained to be bidirectional with two novel prediction tasks, Masked LM and Next Sentence Prediction. The pre-trained BERT model has been shown to give significant improvements in a series of downstream tasks over ELMo and OpenAI GPT (Devlin et al., 2018).

However, identifying offensive language is not a simple task. Challenges during identification include but are not limited to the fact that surface language features fail to capturing subtle semantic difference, and the shortage of undisputed annotated data (Malmasi and Zampieri, 2018). Most of previous studies focus on distinguishing between offensive and non-offensive language (Kwok and Wang, 2013; Djuric et al., 2015), which is the goal of subtask A in the current shared task. But part of challenge consists of the intertwined nature of such messages having negative connotations and profanity. Dinakar et al. (2011) show that it is important to tease these two factors apart. Malmasi and Zampieri (2018) first address the issue of distinguishing hate speech from general profanity.

3 Methodology and Data

The subtasks in the shared task are rather different. In subtask A, the goal is to identify offensive tweets; in subtask B and C, the aim is to distinguish targeted and untargeted offense and to classify the targeted ones into different types. Subtask A requires sensitivity to subtle changes in word meaning in context while the other subtasks are more categorical in nature. However, both suffer from data sparsity. Therefore, we decided, backed by empirical validation on the trial data, to utilize different methods for the subtasks, namely, BERT embeddings for subtask A, and an SVM classifier for subtasks B and C.

The data collection method used to compile the dataset in OffensEval is described by Zampieri et al. (2019a). We used the official training data and trial data provided by the shared task to train the classifier. Our implmentations can be found at: https://github.com/zytian9/SemEval-2019-Task-6.

3.1 Subtask A: Identifying Abusive Content

The goal of subtask A is to identify whether a tweet contains offensive content by training a model to perform binary classification. There are 13,240 tweet instances in the training data, in which each instance has been labeled as either 'offensive' ('OFF') or 'not offensive' ('NOT'). The model takes a tweet as input and predicts the corresponding label of that tweet. We used the trial data as development data.

3.1.1 Model Details

For subtask A, we trained a classifier by fine-tuning a pre-trained BERT Transformer (Devlin et al., 2018) with a linear layer for text sequence classification on top.

The input sentences[1] were first tokenized with the BERT basic tokenizer to perform punctuation splitting, lower casing and invalid characters removal. Then this was followed by WordPiece tokenization (Wu et al., 2016) to split words into subword units, in accordance with the original BERT approach (Devlin et al., 2018). The maximum sequence length was defined as 80, with shorter sequences padded and longer sequences truncated to this length. The order of the input sequence was represented by the learned positional embeddings. The input representation for each tweet is the sum of these token, segment, and position embeddings. As only one sentence serves as input, only the sentence A embeddings are used as the segment embeddings (Devlin et al., 2018).

We selected the $\text{BERT}_{\text{base-uncased}}$ as the underlying BERT model. The $\text{BERT}_{\text{base}}$ consists of 12 Transformer blocks, 12 self-attention heads, and 768 hidden dimension with a total parameters of 110M. It was trained on the BookCorpus (800M words) and the English Wikipedia (2,500M

[1]In BERT, a "sentence" can be a text sequence of arbitrary length. In our case, a "sentence" refers to a tweet even if it may span multiple linguistic sentences.

words). Though the BERT$_{\text{large}}$ model was reported to outperform the BERT$_{\text{base}}$ in a variety of tasks, training and fine tuning BERT$_{\text{large}}$ was too computationally intensive given the time limit. Thus we used BERT$_{\text{base}}$ for accelerated training. The BERT$_{\text{base}}$ model includes a special classification embedding `[CLS]` at the beginning of every sentence, and this token in the final layer was extracted as the aggregate sequence representation for the current classification task. Then a linear layer of 768 dimensions was added on top of BERT$_{\text{base}}$, using the `[CLS]` embeddings of the whole input sequence to predict a binary label. Binary cross-entropy was used as the loss function to fine-tune the classifier.

3.1.2 Implementation

The neural network was implemented in PyTorch (Paszke et al., 2017), and we used the tokenizer, pretrained WordPiece, and positional embeddings and pre-trained BERT from the library `pytorch-pretrained-bert`[2]. Following the recommendation for fine-tuning in the original BERT approach (Devlin et al., 2018), we trained our classifier with a batch size of 32 for 2 epochs. The dropout probability was set to 0.1 for all layers. Adam optimizer was used with a learning rate of 2e-5. The training was carried out on an Nvidia 1070Ti GPU; only taking about 6 minutes in total.

3.2 Subtask B: Categorizing Offense Types

For subtasks B and C, we adopted an SVM classifier. For these two tasks, the BERT classifier performed close to the baseline on the trial data. This could be caused by the limited amount of the training data for these two tasks or inappropriate selection of hyperparameters. Thus, we built a linear SVM classifier to identify the offense type and target.

Subtask B requires the distinction between targeted and untargeted offense. We used an SVM classifier with selected character n gram features for subtask B. For the trial data of subtask B, the classifier achieved a macro F_1 score of 0.5333 and accuracy of 0.5714; both of them considerably higher than the baseline. But since the labels of two classes were accidentally flipped in our submission, our results were not competitive. We also reconstructed test F_1 from the flipped confusion matrix. If the labels were not flipped, the test F_1 should be 0.5946.

3.3 Subtask C: Identifying the Target of Abuse

Subtask C requires the classifier to identify three types of offense target, 'Individual' ('IND'), 'Group'('GRP') and 'Other'('OTH'). The training set is rather imbalanced: The minority class OTH constitutes around 10 percent of all the instances, and only occurs once in the trial data. We originally were planning to use the same approach as for subtasks A. However, experiments on the trial data showed a weak performance. For this reason, we decided to use a linear SVM classifier to identify the offense target with three sub-classes since previous studies indicate that SVM classifiers perform well on classification tasks and at par with deep neural networks when features are well selected (Founta et al., 2018; Kumar et al., 2018). For this classifier, we used the Scikit-learn (Pedregosa et al., 2011) implementation, and we used only the training data provided by the shared task.

3.3.1 Model Details

Given that character-level n-gram could reduce the effect of spelling errors and variations in tweets (Schmidt and Wiegand, 2017), we used a bag of character n-grams (with n ranging from 2 to 7 characters) as features in order to characterize the users' language features as robustly as possible. Since in subtask C, we need to identify different types of offense target, we assume that named entity information will be effective for identifying target types. Named entities information was extracted by spaCy, which is based on the entity types from OntoNotes 5 corpus[3]. Given that this task aims to identify three types of targets, namely individual, group and other, we used the named entity information by classifying all the entity types into three major types and counting the number of each type separately. The first type only includes PERSON entities, the second type consists of entity types related to a group sense, for example ORG, NORG, and GPE, and the last type includes all occurrences of the other entity types.

Nobata et al. (2016) found that linguistic features such as tweet length, average word length,

[2]https://github.com/huggingface/pytorch-pretrained-BERT

[3]`https://spacy.io/api/annotation#named-entities`

System	F_1 macro	Accuracy
All OFF baseline	0.2182	0.2790
All NOT baseline	0.4189	0.7209
$\text{BERT}_{\text{base-uncased}}$	**0.8136**	0.8570

Table 1: The official UM-IU@LING result for subtask A, in comparison to the baselines.

System	F_1 macro	Accuracy
All OFF baseline	0.1934	0.2399
All NOT baseline	0.4319	0.7601
$\text{SVM}_{\text{character-ngram}}$	0.8267	0.8782
$\text{BERT}_{\text{base-uncased}}$	**0.8388**	0.8722
$\text{BERT}_{\text{base-cased}}$	0.8094	0.8500
$\text{BERT}_{\text{base-multiling-unc.}}$	0.4300	0.7625
$\text{BERT}_{\text{base-multiling-cased}}$	0.8179	0.8718

Table 2: Results on the trial data for subtask A.

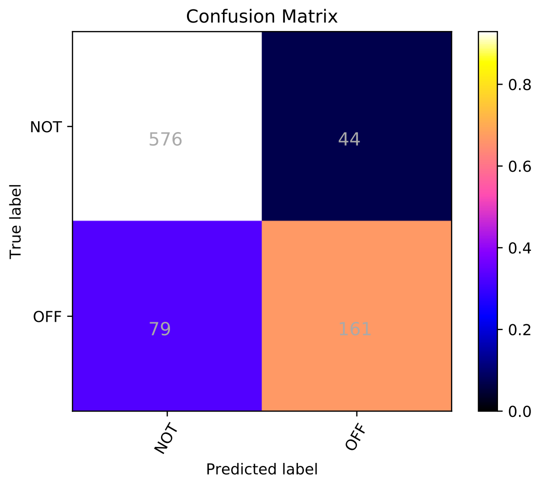


Figure 1: The UM-IU@LING confusion matrix for subtask A.

number of punctuation, number of discourse connectives can be useful for detecting abusive language. In this study, we adopt 9 features from their work. Besides the n-gram features, named entity, and linguistic features, we also adopted emoji and emoticons as additional features, which have been shown to be useful in sentiment analysis tasks (Kouloumpis et al., 2011; Shiha and Ayvaz, 2017). Emoticons are extracted using the s regular expressions by C. Potts[4]. We also added three emoji sentiment features, which consist of the positive, negative, and overall sentiment scores based on the Emoji Sentiment Ranking (Novak et al., 2015).

We performed feature selection for the n-gram features using a filtering approach with information gain, which has proven to be effective in social media sentiment classification (Kübler et al., 2018).

Our final submission is a linear SVM classifier (C=0.1, squared-hinge loss function) with 1000 selected character n-grams of length 2-7. Adding linguistic and emoji features resulted in small gains on the trial data and was that not considered useful for the official version.

4 Results

4.1 Subtask A

Our best result for subtask A along with the official baselines are summarized in Table 1. The BERT classifier achieved a macro F_1 score of 0.8136, clearly exceeding the baseline of 0.4189 and ranking the system 3rd out of 103 submissions. This demonstrates that our model can effectively identify whether a given tweet contains offensive content or not. The confusion matrix in Figure 1 further illustrates the error pattern of our classifier, which more often misclassified offensive tweets as being not offensive. One explanation of the results may be the classifier's preference for the majority class. But it is possible that our classifier may not capture some of the subtle nuances in meaning and contexts. However, the results also show that the macro F_1 score is only about 4.5 percent points lower than the accuracy (0.8136 vs. 0.8570). This is a clear indication that the classifier is successful in modeling the minority class of offensive tweets.

4.1.1 Ablation Analysis

We performed an ablation analysis on our BERT classifier using the training and the trial data. First, we retrained the classifier by varying the learning rate. The macro F_1 dropped to the baseline of 0.4318 with a learning rate of either 2e-8 or 2e-3, which indicates that the system is sensitive to change in learning rates.

The selection of sequence length only has a minimal influence on the final performance, with a tendency for longer sequence length to improves prediction accuracy: Setting the input sequence length to 60 reduces the macro F_1 minimally to 0.8212, and decreasing the input length to 40 decreases the macro F_1 to 0.8126.

[4]http://sentiment.christopherpotts.net/tokenizing.html #emoticons

ID	Tweet	Label	Prediction
50	okay but it actually sucks so much that the first year I COULD go to every Reeperbahn Festival day, I'm in Strasbourg and can only attend the last day	NOT	OFF
263	My mom just called me and said she is joining the NFL boycott. How many of yall are with us? F that league #NFLBoycott	OFF	NOT
126	@User @User @User They don't. The GOP will keep supporting racketeer, illegitimate Trump. They never will stop the corruption of tRump. They are in it for the money. They want to destroy American democracy.	NOT	OFF

Table 3: Misclassified examples for subtask A from the trial data. Usernames are anonymized.

There are several versions of pre-trained $\text{BERT}_{\text{base}}$[5]. We compared the performance of these different versions of $\text{BERT}_{\text{base}}$ and the results are summarized in Table 2. Generally, these variants of $\text{BERT}_{\text{base}}$ tend to give similar performance but $\text{BERT}_{\text{base-uncased}}$ achieved the best performance on the trial data. It is unclear why $\text{BERT}_{\text{base-multilingual-uncased}}$ did not learn to perform the task beyond the baseline. Additional hyperparameter tuning might be necessary in this case. Overall, these results demonstrate that though BERT can give superior performance in detecting hate speech, it is somewhat sensitive to the change of hyperparameters. We also find that the SVM classifier achieved a higher accuracy on the trial data, but there is a significant drop in macro F_1 when compared with the BERT model. This shows that the BERT model performs better on the minority class.

4.1.2 Error Analysis

We show examples of misclassified tweets in Table 3. In example 263, the BERT classifier failed to identify the offensive word "F". It is common for people to use euphemisms to tone down swear words in certain situations. The classifier could miss these word variants, especially when the word variant is the only offensive word in the given tweet. For tweet 50, the word "sucks" is the only word that is often used offensively. However, the given tweet is not offensive because the author only describes their mood instead of insulting someone else. These misclassifications seem to indicate that the classifier reacts to trigger words with negative connotations but may not be capable of interpreting the words with respect to the larger context.

When examining the prediction errors, we consistently noticed that the BERT classifier is highly effective in identifying tweets with words that are negative or offensive in most linguistic contexts. The real challenge is that not all tweets containing negative or potentially insulting words are offensive; there are subtle differences between a negative opinion and an insult towards someone. However, the model cannot distinguish these subtle differences in meaning in the proper cultural or socio-political contexts. Additionally, it is not robust enough to detect swear word variants or atypical spellings common in social media.

[5]https://github.com/google-research/bert

System	**F_1 macro**	**Accuracy**
All GRP baseline	0.1787	0.3662
All IND baseline	0.2130	0.4695
All OTH baseline	0.0941	0.1643
SVM classifier	**0.5243**	0.6854

Table 4: The official UM-IU@LING results (SVM) for subtask C.

System	**$F1_1$ macro**	**Acc.**
All IND baseline	0.3041	0.8387
All GRP baseline	0.0762	0.1290
All OTH baseline	0.0208	0.0323
$\text{SVM}_{\text{character-ngram}}$	0.3915	0.8065
$\text{SVM}_{\text{word-ngram}}$	0.3554	0.6774
$\text{SVM}_{\text{char+ling+emoji}}$	**0.3971**	0.8065
$\text{SVM}_{\text{char+ling+emoji+entity}}$	0.3901	0.7742

Table 5: Results on the trial data for subtask C.

4.2 Subtask C

Table 4 shows our best result for subtask C in comparison to the official baselines. The macro F_1 score of the SVM classifier is 0.5243, which is considerably higher than the baseline and ranks the system 27th out of 65 submissions. The confusion matrix in Figure 2 indicates that our classi-

ID	Tweet	Label	Prediction
17	@User Obama fed the country shit sandwiches for 8 years. Maybe Jim just has his addled mind confused about dates and who fed who what..	GRP	IND
22	@User The Catholic Church is really screwed up. Nothing new here.	GRP	OTH
31	@User Yeah thanks to your Nobel Emmy award winning idiot chief flip flopping on everything from Iran to gun control.	IND	GRP

Table 6: Misclassified examples from the trial data for subtask C.

fier performed well on identifying the IND class, was effective for the GRP class, but often failed to distinguish the OTH class from the other two classes. This clearly shows that the sparsity of training data for the minority class OTH affects the performance of our classifier negatively.

The performance of the classifier with different features is shown in Table 5. Since there are only 31 instances in the trial set and it is rather imbalanced, we can see that the highest accuracy is reached by classifying all examples as IND, i.e., the all IND baseline. Even though none of the classifiers outperformed the baseline in terms of accuracy, all the classifiers achieved significantly higher macro F_1 scores, which shows that they are better at identifying the other two classes. After adding linguistic and emoji features, the character n-gram model showed a slight improvement in macro F_1 score and achieved the highest accuracy along with the simple character n-gram model. But both macro F_1 and accuracy dropped when entity information was added.

Table 6 presents examples of misclassified tweets in the trial set. In example 17, two persons are mentioned, “Obama” and “Jim’, and both of them are insulted, however not as a group but individually. The classifier labeled this example as IND. In example 22, the classifier is misguided by the word ’Church’ and wrongly classifies it as OTH. Example 31 is similar to example 17. Here, there are two potential targets, ’Nobel Emmy award winning idiot’ and ’Iran’ that could trigger the group sense, which significantly affects the classifier’s judgment.

The errors analysis indicates that the classifier has the ability to distinguish individual and group targets, but it fails to capture the relation between different entities and sometimes misidentifies the target category of offensive language.

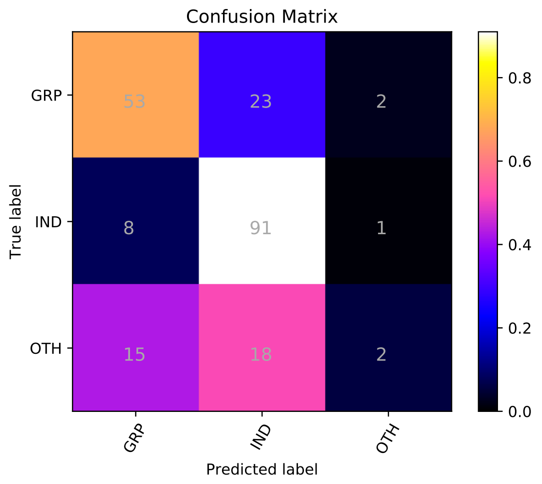


Figure 2: Confusion matrix for the SVM classifier for subtask C.

5 Conclusion

In this study, we report our systems for OffensEval subtasks A and C. In subtask A, we trained a neural network based classifier by fine-tuning the pre-trained $\text{BERT}_{\text{base}}$ model to detect offensive tweets. In subtask C, we used a linear SVM with character n-gram features to identify the target of hate speech.

The evaluation results indicate that our system is capable of detecting offensive language robustly, and it has a good chance of identifying the target. However, there is room for improvement. In the future, in order to capture subtle meaning and overcome the data sparsity, we plan to take syntactic and semantic features into consideration and investigate the combination of selected surface features and pre-trained word embeddings.

References

Ignacio Arroyo-Fernández, Dominic Forest, Juan-Manuel Torres-Moreno, Mauricio Carrasco-Ruiz, Thomas Legeleux, and Karen Joannette. 2018. Cyberbullying detection task: The EBSI-LIA-UNAM system (ELU) at COLING’18 TRAC-1. In *Proceed-*

ings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018), pages 140–149.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760, Perth, Australia.

Pete Burnap and Matthew L Williams. 2015. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet*, 7(2):223–242.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of ICWSM*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Karthik Dinakar, Roi Reichart, and Henry Lieberman. 2011. Modeling the detection of textual cyberbullying. In *The Social Mobile Web*, pages 11–17.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pages 29–30.

Antigoni Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large scale crowdsourcing and characterization of twitter abusive behavior. In *Twelfth International AAAI Conference on Web and Social Media*.

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. 2011. Twitter sentiment analysis: The good the bad and the OMG! In *Fifth International AAAI Conference on Weblogs and Social Media*.

Sandra Kübler, Can Liu, and Zeeshan Ali Sayyed. 2018. To use or not to use: Feature selection for sentiment analysis of highly imbalanced data. *Natural Language Engineering*, 24(1):3–37.

Ritesh Kumar, Atul Kr Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 1–11.

Irene Kwok and Yuzhou Wang. 2013. Locate the hate: Detecting tweets against blacks. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in discriminating profanity from hate speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*, pages 145–153.

Petra Kralj Novak, Jasmina Smailović, Borut Sluban, and Igor Mozetič. 2015. Sentiment of emojis. *PloS One*, 10(12).

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/languageunsupervised/languageunderstandingpaper.pdf*.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia, Spain.

Mohammed Shiha and Serkan Ayvaz. 2017. The effects of emoji in sentiment analysis. *International Journal of Computer and Electrical Engineering (IJCEE.)*, 9(1):360–369.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin

Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

USF at SemEval-2019 Task 6: Offensive Language Detection Using LSTM With Word Embeddings

Bharti Goel and Ravi Sharma
Department of Computer Science and Engineering, University of South Florida, USA
Emails: bharti@mail.usf.edu, ravis@mail.usf.edu

Abstract

In this paper, we present a system description for the SemEval-2019 Task 6 submitted by our team. For the task, our system takes tweet as an input and determine if the tweet is offensive or non-offensive (Sub-task A). In case a tweet is offensive, our system identifies if a tweet is targeted (insult or threat) or non-targeted like swearing (Sub-task B). In targeted tweets, our system identifies the target as an individual or group (Sub-task C). We used data pre-processing techniques like splitting hashtags into words, removing special characters, stop-word removal, stemming, lemmatization, capitalization, and offensive word dictionary. Later, we used keras tokenizer and word embeddings for feature extraction. For classification, we used the LSTM (Long short-term memory) model of keras framework. Our accuracy scores for Sub-task A, B and C are *0.8128*, *0.8167* and *0.3662* respectively. Our results indicate that fine-grained classification to identify offense target was difficult for the system. Lastly, in the future scope section, we will discuss the ways to improve system performance.

1 Introduction

In recent years, there has been a rapid rise in social media platforms and surge in the number of users registering in order to communicate, publish content, showcase their skills and express their views. Social media platforms like Facebook and Twitter have millions of registered users influenced by the countless user-generated posts on daily basis (Zeitel-Bank and Tat, 2014). While on one hand social media platforms facilitate the exchange of views, effective communication and can be seen as a helping mode in crisis. On the other hand, they open up the window for anti-social behavior such as bullying, stalking, harassing, trolling and hate speech (Malmasi and Zampieri, 2018; Wiegand et al., 2018; ElSherief et al., 2018; Zhang et al., 2018). These platforms provide the anonymity and hence aid users to indulge in aggressive behavior which propagates due to the increased willingness of people sharing their opinions (Fortuna and Nunes, 2018).

This aggression can lead to foul language which is seen as "offensive", "abusive", or "hate speech", terms, which are used interchangeably (Waseem et al., 2017). In general, offensive language is defined as derogatory, hurtful/ obscene remarks or comments made by an individual (or group) to an individual (or group) (Wiegand et al., 2018; Baziotis et al., 2018). The offensive language can be targeted towards a race, religion, color, gender, sexual orientation, nationality, or any characteristics of a person or a group. Hate Speech is slowly plaguing the social media users with depression and anxiety (Davidson et al., 2017; Zhang et al., 2018), which can be presented in the form of images, text or media such as audio, video, etc. (Schmidt and Wiegand, 2017).

Our paper presents the data and task description followed by results, conclusion and future work. The purpose of Task 6 is to address and provide an effective procedure for detecting offensive tweets from the data set provided by shared task report paper (Zampieri et al., 2019b). The shared task is threefold. The Sub-task A ask us to identify whether the given tweet is offensive or non-offensive. In Sub-task B offensive tweets are to be classified as targeted (person/group) or non-targeted (general). Sub-task C ask us to do classification of the offensive tweets into individual, group or others. We apply the LSTM with word embeddings in order to perform the multi-level classification.

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 796–800
Minneapolis, Minnesota, USA, June 6–7, 2019. ©2019 Association for Computational Linguistics

2 Related Work

Technological giants like Facebook, Google, YouTube and Twitter have been investing a significant amount of time and money towards the detection and removal of offensive and hate speech posts that give users a direct or indirect negative influence (Fortuna and Nunes, 2018). However, lack of automation techniques and ineffectiveness of manual flagging has lead to a lot of criticism for not having potent control for the problem (Zhang et al., 2018). The process of manual tagging is not sustainable or scalable with the large volumes of data exchanged in Social media. Hence, the need of the hour is to do automatic detection and filtering of offensive posts to give the user quality of service (Fortuna and Nunes, 2018).

The problem of automatic Hate Speech Detection is not trivial as offensive language may or may not be meant to insult or hurt someone and can be used in common conversations. Different language contexts are rampant in social media (Davidson et al., 2017). In recent years, linguistics, researchers, computer scientists, and related professionals have conducted research towards finding an effective yet simple solution for the problem. In papers, (Schmidt and Wiegand, 2017; Fortuna and Nunes, 2018), authors survey the state of the art methods along with the description of the nature of hate speech, limitations of the methods proposed in literature and categorization of the hate speech. Further, authors mainly classified the features as general features like N-grams, Part-Of-Speech (POS) and sentiment scores and, specific features such as othering language, the superiority of in-group and stereotype. In (Silva et al., 2016), authors list categories of hate speech and possible targets examples.

The research carried over the years has employed various features and classification techniques. The features include the bag of words (Greevy and Smeaton, 2004; Kwok and Wang, 2013), dictionaries, distance metrics, N-grams, IF-IDF scores, and profanity windows (Fortuna and Nunes, 2018). Authors in (Davidson et al., 2017) used a crowdsourced hate speech lexicon to collect tweets and train a multi-class classifier to distinguish between hate speech, offensive language, and non-offensive language. The authors in paper (Waseem et al., 2017) presents a typology that gives the relationship between various sub-tasks such as cyber-bullying, hate speech, offensive, and online abuse. They synthesize the various literature definitions and contradiction together to emphasize the central affinity among these sub-tasks. In (Gambäck and Sikdar, 2017), authors used a Convolutional Neural Network model for Twitter hate speech text classification into 4 classes: sexism, racism, both sexism-racism and neither. Similar approaches using deep learning have been employed in (Agrawal and Awekar, 2018; Badjatiya et al., 2017) for detecting hate speech and cyberbullying respectively.

Authors in (Zhang et al., 2018) have proposed Deep Neural Network (DNN) structures which serve as a feature extractor for finding key semantics from hate speech. Prior to that, they emphasize the linguistics of hate speech, that it lacks discriminative features making hate speech difficult to detect. Authors in (ElSherief et al., 2018) have carried out the analysis and detection of the hate speech by classifying the target as directed towards a specific person/identity or generalized towards a group of people sharing common characteristics. Their assessment states that directed hate consists of informal, angrier and name calling while generalized hate consists of more religious hate and use of lethal words like murder, kills and exterminate.

3 Problem Statement

In Task 6, three level offensive language identification is described as three Sub-tasks A, B and C. For Sub-task A, tweets were identified as offensive (tweet with any form of unacceptable language, targeted/non-targeted offense) or not offensive. For Sub-task B, the offensive tweets are further categorized into targeted or non-targeted tweets. The targeted offense is made for an individual or group while the untargeted offense is a general use of offensive language. Later for Sub-task C, targeted tweets are further categorized according to the target, individual, group or others. This step by step tweet classification will lead to the detailed categorization of offensive tweets.

4 Data

The data collection methods used to compile the dataset used in OffensEval are described in Zampieri et al. (2019a). The OLID dataset collected from Twitter has tweet id, tweet text, and labels for Sub-task A, B, and C. We have also used Offensive/Profane Word List (Ahn, 2019) with *1,300+* English terms that could be found offensive for lexical analysis of tweets to see check probability of tweet being offensive if a tweet has an offensive word.

4.1 Data Pre-processing

The data is raw tweet data from Twitter and hence data cleaning and pre-processing is required. We used the following steps for data pre-processing:

1. Split hashtags over Capital letters: In this step hashtags are divided into separate words (for example, "#TweetFromTheSeat" will be converted to "Tweet from the seat"). Generally, while writing hashtags multiple words are combined to form single hashtag, where each word is started with a capital letter. Here, we take advantage of this property of hashtag and generate a string of words from it.

2. Remove special characters: In this step we removed all special characters from the tweet and resultant tweet will contain only alphabets and number. In Twitter domain "#" is important special character. Splitting of hashtags,"#Text" into "#" and "Text", retains the purpose of the hashtags after removal of "#" . Other special characters (for e.g. ",", ".","!", etc) are not much informative for given context.

3. Removal of stop-words, Stemming and Lemmatization: In this step we used NLTK (Loper and Bird, 2002) list of stop-words to remove stopwords, classic Porter Stemmer (Porter, 1980) for stemming and NLTK (Loper and Bird, 2002) Word Net Lemmatizer for lemmatization.

4. Capitalization: This is the last step for data pre-processing and all characters are converted to capital letters. In Twitter domain uppercase characters are said to portray expression, but this is not true for all cases. Also, keeping cases intact may lead to over-fitting during training.

5. Embedding "Offensive": This is an optional step. We used offensive word list (Ahn, 2019) to find offensive words in the tweet. Later for tweets with matched offensive words were embedded with "Offensive" as a word.

5 System Description

We used a four-layer neural network with each layer detailed below:

System	F1 (macro)	Accuracy
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
LSTM (5,0.2)	**0.7382**	**0.8128**

Table 1: Results for Sub-task A.

System	F1 (macro)	Accuracy
All TIN baseline	0.4702	**0.8875**
All UNT baseline	0.1011	0.1125
LSTM (5,0.2)	0.5925	0.8167
LSTM (20,0.2)	0.5291	0.6125
LSTM (20,0.2) and word list	**0.6171**	0.7667

Table 2: Results for Sub-task B.

The first layer of our network is an embedding layer, which takes tokens as inputs (each sentence is converted into index sequences using tokenizer). This layer convert sequences to dense vector sequences generating embedding table used by the next layer. We used tokenizer for top *1000* words and embedding dimension of *128* for our system. The second layer is SpatialDropout1D layer which helps promote independence between feature maps. We used *0.1* as rate or Fraction of the input units to drop. This layer is mainly used to covert multi-dimensional input to one-dimensional input using dropout method.

Third layer is LSTM (Hochreiter and Schmidhuber, 1997) layer with dropout and recurrent dropout as *0.2*. This layer serves as a recurrent neural network layer which was only for short term memory. LSTM (long short-term memory) takes care of longer dependencies. The dimension of LSTM hidden states is *200* for our system. Finally, we used a dense layer with Softmax function for binary classification in-case of Sub-task A and B, and three class classification for Sub-task C. The dimension of the dense layer is *200*.

For hyperparameter selection, we used different train and validation splits. The batch size is *64*, and the maximal training epoch is varied with different system ranging from *5* to *50* (Performance was decreasing for higher epochs). We used RMSProp as the optimizer for network training. The performance is evaluated by macro-averaged F1-score and Accuracy by task organizers.

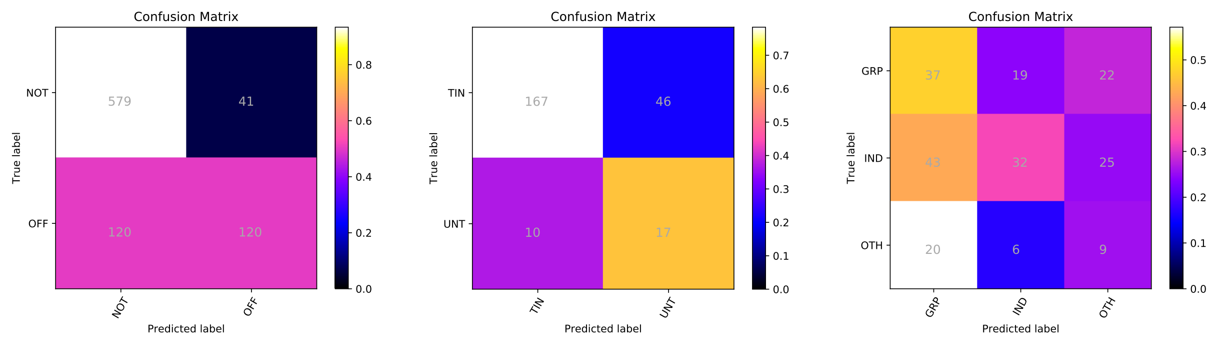


Figure 1: a) Sub-task A, LSTM (epoch=5, dropout= 0.2), b) Sub-task B, LSTM (dropout=0.2, epochs=20), c) Sub-task C, LSTM(dropout=0.2, epochs=50)

System	F1 (macro)	Accuracy
All GRP baseline	0.1787	0.3662
All IND baseline	0.2130	**0.4695**
All OTH baseline	0.0941	0.1643
LSTM (20,0.2) and word list	0.1849	0.1878
LSTM (50,0.2)	**0.3404**	0.3662
LSTM (50,0.2) and word list	0.2832	0.3521

Table 3: Results for Sub-task C.

6 Results

Tables 1, 2 and 3 shows F1 (macro) and Accuracy scores for the submitted systems. We can see that for all the sub-tasks our system has F1 (macro) and as described in (Zampieri et al., 2019a) F1 (macro) is used for performance analysis by task coordinators. Best results are highlighted in bold ink in tables and confusion matrix for them is also shown in Figure 1 for Sub-tasks A, B and C. In **Sub-task A** we achieved *0.8128* and *0.7382* as accuracy and F1 respectively. For this task we submitted only one system with LSTM network dropout *0.2* and *5* epochs. For **Sub-task B** we submitted three runs but, the best performance is achieved by system with LSTM dropout of *0.2*, *20* epochs and offensive word list described in Section 4. Later in **Sub-task C** we submitted three runs and best performance was LSTM with *50* epochs and *0.2* dropout.

7 Conclusion

In this paper we used LSTM network to identify offensive tweets and categorize offense in subcategories as described in Section 3 for Task 6: Identifying and Categorizing Offensive Language in Social Media. We used an embedding layer followed by LSTM layer for tweet classification. Three tasks of OffensEval Sub-task A, B, and C were of varied difficulty level. The main reason can be decreasing amount of data for each of them, where Sub-task A has more data followed by Sub-task B categorizing offensive tweets identified by Sub-task A and, Sub-task C categorizing targeted offense identified by Sub-task B. Data was also unbalanced leading to more importance for majority class but after applying cost function we found that accuracy was decreased with increased errors in identification of majority class.

8 Future Work

For future work, we would like to use additional datasets like TRAC-1 data (Kumar et al., 2018), (Davidson et al., 2017), and would collect data from Twitter to get diverse data. To be consistent with substantial research done in recent years we want to employ a combination of textual features like the bag of words n-grams, capitalized characters, sentiment scores, e.t.c. Also, we want to focus more on specific features like semantics and linguistic features intrinsic to hate/offensive rather than just generic text-based features. For that, we want to use character level deep LSTM which can be used to extract the semantic and syntactic information. Finally, we want to explore more about the similarities and dissimilarities between the profanity and hate speech, establishing more profound way of extracting features in order to make the detection system more responsive.

References

Sweta Agrawal and Amit Awekar. 2018. Deep learning for detecting cyberbullying across multiple social media platforms. *CoRR*, abs/1801.06482.

Luis von Ahn. 2019. Offensive/profane word list. *Carnegie Mellon School of Computer Science*.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, pages 759–760, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Baziotis, Christos, Athanasiou, Nikos, Athanasia, Paraskevopoulos, Georgios, Ellinas, Nikolaos, Alexandros, and et al. 2018. Ntua-slp at semeval-2018 task 3: Tracking ironic tweets using ensembles of word and character level attentive rnns.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Edel Greevy and Alan F. Smeaton. 2004. Classifying racist texts using a support vector machine. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 468–469, New York, NY, USA. ACM.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Irene Kwok and Yuzhou Wang. 2013. Locate the hate: Detecting Tweets Against Blacks. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

M.F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Silva, Leandro, Mondal, Correa, Denzil, Benevenuto, Fabricio, Weber, and Ingmar. 2016. Analyzing the targets of hate in online social media.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Natascha Zeitel-Bank and Ute Tat. 2014. *Social Media and Its Effects on Individuals and Social Systems*, Human Capital without Borders: Knowledge and Learning for Quality of Life; Proceedings of the Management, Knowledge and Learning International Conference 2014, pages 1183–1190. ToKnowPress.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

UTFPR at SemEval-2019 Task 6: Relying on Compositionality to Find Offense

Gustavo Henrique Paetzold
Universidade Tecnológica Federal do Paraná / Toledo-PR, Brazil
ghpaetzold@utfpr.edu.br

Abstract

We present the UTFPR system for the OffensEval shared task of SemEval 2019: A character-to-word-to-sentence compositional RNN model trained exclusively over the training data provided by the organizers. We find that, although not very competitive for the task at hand, it offers a robust solution to the orthographic irregularity inherent to tweets.

1 Introduction

Text classification tasks can take a wide variety of forms, and some of them, such as sentiment and emotion analysis, have managed to grab a lot of attention from researchers in recent years. More recently, however, the public's growing engagement in debates on the topics of free speech and politics has led the Natural Language Processing (NLP) and Machine Learning (ML) communities to take an interest in classification tasks related to identifying and categorizing patterns of profanity, hate speech and offense. The prominence of shared tasks held on these topics, such as those of Fersini et al. (2018) and Wiegand et al. (2018), are great examples of the research community coming together to attempt to create more reliable solutions for these challenges.

While hate speech is commonly characterized by specific slurs and other offensive expressions that convey prejudice against a certain group or individual, offensive speech is often more challenging to identify because it encompasses a more broad spectrum of language, featuring expressions that do not necessarily convey prejudice (Malmasi and Zampieri, 2018). Technologies that identify these types of patterns could, for instance, help a social media platform on profiling users and take appropriate action whenever someone breaks user agreements and/or terms of use.

Identifying offensive language within the content of social media platforms is particularly challenging, since this type of content is usually littered with irregular orthography, meta-characters, slang and others. Since a lot of the effort from the research community focuses on identifying offensive language in social media platforms, an NLP approach for such a task must be able to overcome those hurdles in some way. Some of the preferred methods for handling the orthographic irregularity of social media content are using word embeddings trained over tweets (Rozental et al., 2018) or regularizing unusual spellings (Bertaglia and das Graças Volpe Nunes, 2017), but neither of them ensure that every possible orthographically irregular word will be understood by the NLP model in question. Recently, however, there have been a lot of contributions that present compositional neural models that learn numerical representations of words based on the sequence of characters that compose them (Kim et al., 2016; Ling et al., 2015; Balazs et al., 2018; Paetzold, 2018). These models have been demonstrated to be both effective in text classification, and robust when faced with orthographic irregularity.

In this paper, we present the UTFPR system submitted to the OffensEval shared task of SemEval 2019, which employs compositional neural models to identify offensive language in tweets. In the following sections we describe the task (section 2), our model (section 3) and experiments (sections 4 to 5).

2 Task Summary

The UTFPR systems described herein are a contribution to the OffensEval shared task held at the SemEval 2019 workshop (Zampieri et al., 2019b). In this shared task, participants were tasked with creating innovative classifiers capable of identifying and categorizing offensive tweets. This shared tasks has 3 sub-tasks:

- **Task A:** Binary classification task that consists in judging whether a tweet is offensive or not.
- **Task B:** Binary classification task that consists in identifying whether or not an offensive tweet was targeted towards a specific person or group.
- **Task C:** Consists in identifying whether an offensive tweet was targeted at a person, group or something else (3-class classification).

We decided to focus our efforts on Task A exclusively. The organizers provided participants a training set with $13,240$ instances, a trial set with 320, and a test set with 860. Each instance is composed of a tweet and its respective labels for tasks A, B and C. The datasets were annotated by humans of undisclosed background (Zampieri et al., 2019a).

3 The UTFPR Model

As we have previously mentioned, ours is a compositional Recurrent Neural Network (RNN) inspired by the ones introduced by Ling et al. (2015) and Paetzold (2018). Our RNN learns word representations based on the sequence of characters that compose them, then learns sentence representations based on the word representations previously learned. Figure 1 illustrates the architecture of our model in detail.

The model takes as input a potentially offensive tweet. It first produces character embeddings for the characters of each word in the sentence, then passes them through a sequence of bidirectional RNN layers in order to produce character-to-word numerical representations for them. These word representations are then passed onto another sequence of bidirectional RNN layers, which in turn produce a single word-to-sentence numerical representation for the sentence. A dense layer connected to a softmax layer produces the final binary class, which can be OFF (for offensive) and NOT (for not offensive).

Because the dataset provided for training is rather small, we suspected that the character-to-word representations produced through this training data would not be reliable enough for the task. Because of that, we decided to train two different model variants:

- **UTFPR-Scratch:** The model depicted in Figure 1 trained from scratch over the shared task's training set exclusively.
- **UTFPR-Reuse:** The same model depicted in Figure 1, except instead of training its character-to-word RNN layers from scratch along with the rest of the model, they are taken from a similar compositional model pre-trained by Paetzold (2018) over a much larger dataset for the Emotion Analysis shared task held at WASSA 2018 (Klinger et al., 2018). The training set of the WASSA 2018 shared task has $153,383$ instances, each composed of a tweet with a target emotion word replaced with a [#TRIGGERWORD#] marker, and an emotion label that could be either joy, sad, disgust, anger, surprise, or fear.

The architecture of our models is identical, and their specifications are:

- **Size of character embeddings:** 15
- **RNN layer type:** Gated Recurrent Units
- **RNN layer depth:** 2 (for all layers sets)
- **RNN layer size:** 60 (for all layers)
- **Dropout proportion:** 25%
- **Loss function:** Cross-entropy
- **Framework used:** PyTorch[1]

We chose to use the PyTorch framework due to the fact that it employs dynamic computational graphs, and hence they do not require us to set a fixed maximum size for the words in the dataset. This feature of PyTorch only allows us to create a much more flexible model that can handle any word size, but also disregards the needs for padding.

To train our models, we split the training set into a training portion ($10,000$ instances) and a development portion ($3,240$ instances). The models were left training for hundreds of iterations, and after each iteration a version of each model was saved. The final selected models were the ones with the lowest attained error on the development

[1]https://pytorch.org

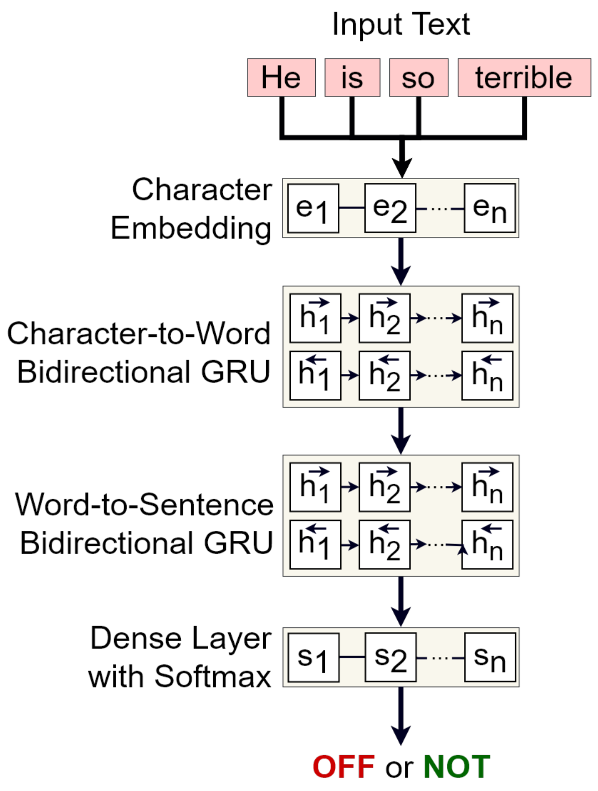


Figure 1: Architecture of the UTFPR system.

portion of the data. We conducted a preliminary evaluation over the trial set to determine which of the variants to submit. The macro-averaged F-scores, which are illustrated in Table 2, show that using pre-trained character-to-word RNN layers actually compromised the performance of our model in this instance, hence we opted to submit the UTFPR-Scratch variant.

System	F-score
UTFPR-Scratch	0.770
UTFPR-Reuse	0.599

Table 1: Macro-averaged F-scores for the trial set

4 Performance on Shared Task

The systems submitted to the shared task were evaluated through their macro-averaged F1-score. The results on Table 2 showcase the results obtained by UTFPR-Scratch, as well as the top 3 and bottom 3 systems submitted to Task A. As it can be noticed, our system did not perform very well, placing 93rd out of 103 teams. The confusion matrix of UTFPR-Reuse in Figure 3 shows that the main reason behind this poor showing was the large amount of false negatives predicted. Upon inspecting the labels predicted, we found that the UTFPR-Scratch system would predict offense mostly for tweets with a lot of profanity and with hashtags associated with the Donald Trump administration, such as “#BuildTheWall”.

Rank	System	F-score
1	pliu19	0.829
2	anikolov	0.815
3	lukez	0.814
93	UTFPR	0.528
101	hamadanayel	0.458
102	magnito60	0.422
103	AyushS	0.171

Table 2: Macro-averaged F-scores for the trial set

5 Robustness Assessment

As we’ve already mentioned, one of the main advantages of compositional RNN models that learn word representations from character sequences is the fact that they handle low-frequency and out-of-vocabulary words in an elegant way, since they are able to produce a numerical representation for any word. Because of that, these models tend to be much more resilient when presented with noisy

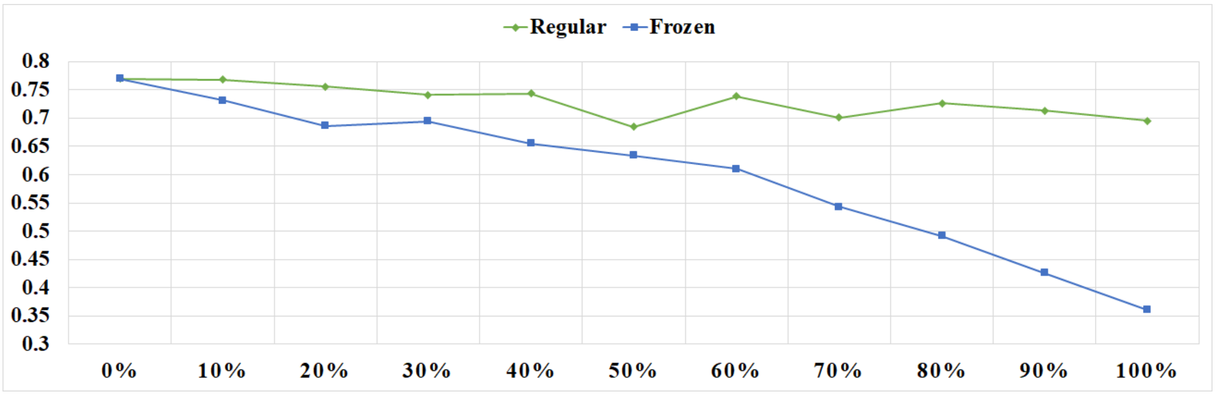


Figure 2: Results of our robustness experiments. The vertical and horizontal axes presents macro-averaged F-scores and the percentage of words with noise introduced to them, respectively. The dots represent the scores obtained by the regular UTFPR-Scratch model and a frozen version that treats all words outside of the training set as out-of-vocabulary words.

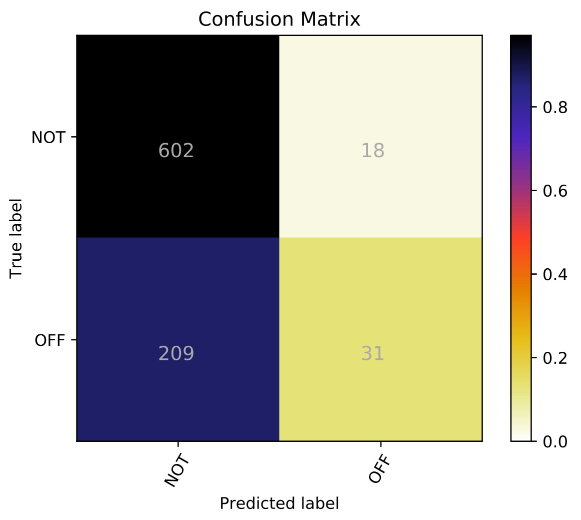


Figure 3: Confusion matrix of the UTFPR-Scratch model on the test set.

input that differs considerably from the input presented during training.

In this experiment, we assess the robustness of our UTFPR models. First, we generated "jammed" versions of the shared task's trial set (since the test set was not made available) with increasing volumes of noise introduced to them. To create a jammed test set, we simply added a noise-inducing modification to $N\%$ randomly selected words of each sentence. The modifications were randomly selected between either removing a randomly selected character form the word (50% chance) or duplicating it (50% chance). We created 11 jammed versions by using $0 \leq N \leq 100$ in intervals of 10. Words with a single character that were subjected to removal were simply discarded from the sentence.

We compared the regular UTFPR-Scratch model (Regular) with a modified version with frozen character-to-word RNN layers (Frozen). The frozen version only produces a numerical representation of a word if it is present in the training set, otherwise, it produces a vector full of 1's signaling an out-of-vocabulary word. The results in Figure 2 show that using the frozen version is much less robust than the regular model, specially when the input sentence has 70% or more of its words out of the training set vocabulary.

6 Conclusions

In this paper we presented the UTFPR system for the OffensEval shared task held at SemEval 2019, which is a compositional RNN model that learns numerical representations of words based on its characters. Our experiments reveal that, although our model is not very competitive for this task specifically (placing 93rd out of 103 participants), it offers a very robust solution to the problem of out-of-vocabulary words. Inspecting the model's output we found that the main cause for its poor performance was the fact that it learned a bias towards the "not offensive" label, which caused it to predict a lot of false negatives. Also, we found that our model was actually better at identifying profanity and controversial topics rather than offense itself. In the future, we intend to explore combining our numerical word representations with richer semantic features in order to train more reliable compositional models for this task.

7 Acknowledgments

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan V GPU used for this research.

References

Jorge Balazs, Edison Marrese-Taylor, and Yutaka Matsuo. 2018. Iiidyt at iest 2018: Implicit emotion classification with deep contextualized word representations. In *Proceedings of the 9th WASSA*, pages 50–56.

Thales Felipe Costa Bertaglia and Maria das Graças Volpe Nunes. 2017. Exploring word embeddings for unsupervised textual user-generated content normalization. *CoRR*, abs/1704.02963.

Elisabetta Fersini, Debora Nozza, and Paolo Rosso. 2018. Overview of the evalita 2018 task on automatic misogyny identification (ami). In *Proceedings of the 6th EVALITA*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of the 2016 AAAI*, pages 2741–2749.

Roman Klinger, Orphee De Clercq, Saif Mohammad, and Alexandra Balahur. 2018. Iest: Wassa-2018 implicit emotions shared task. In *Proceedings of the 9th WASSA*, pages 31–42.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 EMNLP*, pages 1520–1530.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in discriminating profanity from hate speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30(2):187–202.

Gustavo Paetzold. 2018. Utfpr at iest 2018: Exploring character-to-word composition for emotion analysis. In *Proceedings of the 9th EMNLP*, pages 176–181.

Alon Rozental, Daniel Fleischer, and Zohar Kelrich. 2018. Amobee at iest 2018: Transfer learning from language models. In *Proceedings of the 9th WASSA*, pages 43–49.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the germeval 2018 shared task on the identification of offensive language. In *Proceedings of GermEval 2018*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

UVA Wahoos at SemEval-2019 Task 6: Hate Speech Identification using Ensemble Machine Learning

Murugesan Ramakrishnan
Data Science Institue
University of Virginia
mr6rx@virginia.edu

Wlodek Zadrozny
Computer Science
UNC-Charlotte
wzadrozn@uncc.edu

Narges Tabari
Data Science Institute
University of Virginia
ns5kn@virginia.edu

Abstract

With the growth in the usage of social media, it has become increasingly common for people to hide behind a mask and abuse others. We have attempted to detect such tweets and comments that are malicious in intent, which either targets an individual or a group. Our best classifier for identifying offensive tweets for SubTask_A (Classifying offensive vs. non-offensive) has an accuracy of 83.14% and a f1-score of 0.7565 on the actual test data. For SubTask_B, to identify if an offensive tweet is targeted (If targeted towards an individual or a group), the classifier performs with an accuracy of 89.17% and f1-score of 0.5885. The paper talks about how we generated linguistic and semantic features to build an ensemble machine learning model. By training with more extracts from different sources (Facebook, and more tweets), the paper shows how the accuracy changes with additional training data.

1 Introduction

Internet is now accessed by over half of the world's population [1]. In fact, almost 1 million new users are added each day. With social media platforms, people find it a lot easier to get away with the abuse they spew around, in comparison to the offline world. This brings a lot of onus on the Social Network websites to tackle such activities. Majority of the countries have laws to control hate speech which puts tremendous pressure on the concerned websites to curb such activities. Since manual monitoring or defining a specific rule-set might be time consuming, an ensemble machine learning approach has been discussed to avoid complexity and increase interpretability.

The paper focuses on providing solutions to SubTask_A and SubTask_B for the SemEval 2019 competition. Previous works and papers focus on identifying if a tweet is offensive or not. Here, in addition to that, it is identified if an offensive tweet is targeted towards a particular individual or a group *(SubTask_B)*. Such granular information would help the Social Media to make better decisions.

[1] https://news.itu.int/itu-statistics-leaving-no-one-offline/

2 Related Work

This issue has gathered a lot of attention over the past few years with various types of hate speech detection models.

Papers published in the last two years include the surveys by Schmidt and Wiegand (2017) and Fortuna and Nunes (2018) where the authors extract features from the text like sentiment, linguistic features, utilize different lexical resources to tag an offensive tweet, and another paper by Davidson et al. (2017) presenting the Hate Speech Detection data set used in Malmasi and Zampieri (2017) where the authors perform a three way classification - Hate Speech, Offensive and None. By classifying these, the authors talk about specific patterns related to offensive terms. It is found that the usage of cuss words like b*tch and n*gga is fond in both offensive and casual setting, while f*ggot and n*gger were predominantly used in hateful contexts. One of the major takeaways was that lexical methods are effective to identify potentially offensive terms, but are inaccurate at identifying hate speech Other work include: ElSherief et al. (2018); Gambäck and Sikdar (2017); Zhang et al. (2018).

A proposal of typology of abusive language sub-tasks is presented in Waseem et al. (2017) where the author talks about how an offensive tweet can be categorized into four segments - Explicit, Implicit, Directed and Generalized abuse.

These help in creating segment wise features to capture them separately. Finally, methods in identifying profanity vs. hate speech is talked by Malmasi and Zampieri (2018). This work highlighted the challenges of distinguishing between profanity, and threatening language which may not actually contain profane language.

The description of the current task is presented in detail in Zampieri et al. (2019b), which clearly provides the context and underlying problem statement for this paper.

3 Data

For this project, the data set provided by the organizers of OffensEval 2019 was used.The data collection methods used to compile the data set used is described in Zampieri et al. (2019a). The data set consisted of 13,241 records of training observations with the following types of response variables : 1) If a tweet is offensive or not 2) If an offensive tweet is targeted towards an individual (IND) or a group (GRP).

To validate if the performance would increase, an additional data source was also used. The main hypothesis behind including the data was that more data would result in a better accuracy. So, the data set that closely aligned with the current objective was considered for the analysis. This data set was used as a part of the competition organized by TRAC [2]. This contained the response variable with the categories - 'Covertly Aggressive', 'Overtly Aggressive' and 'Non Aggressive'. To maintain consistency with the current data set, 'Covertly Aggressive' and 'Overtly Aggressive' were tagged as 'Offensive' and the rest as 'Not Offensive'.

Including both the data sets, there were a total of 25,239 observations.

The distribution of variables of the original data set is as follows,

SubTask_A: Offensive (33%), and Non-offensive (67%). **SubTask_B:** Out of the 33% offensive tweets, it is seen that there are Targeted (88%), and Untargeted (12%)

4 Methodology

The Methodology involved two sub-works - Feature Engineering and Ensemble Model building.

[2] https://sites.google.com/view/trac1/shared-task/

Various features were extracted to get the semantics of the words and tweets.

4.1 Feature Engineering

Character n-grams
Inspired from earlier works, character n-grams were used especially to tackle misspelled words or words without spaces like 'fu*koff' and 'fu*kasdf'. In both of these cases, character 4-gram would detect the sub-word 'fu*k'.

Word n-grams
Apart from using just 1-gram, 3-gram and 4-gram really helped in identifying the context of the tweet and focus on words like *'not good'* where 'not' negates the next word.

Cuss-word Dictionary and Profanity Checker
A list of cuss words were scraped from *www.noswearing.com.* This helped in identifying such cuss words in tweets which occurred only once or twice in the whole corpus. Profanity checker libraries like *profanity* were also used along with the scraped list. These helped in creating features like *cuss-word count* and *cuss-word position*.

GloVe Embedding
The use case of GloVe embedding were two-fold. One, average embedding could be found for a tweet which can then be used as a feature space. Two, once the top-30 features were obtained from the initial training, GloVe model was used to find most similar words to them thereby creating a feature representing potential offensive terms.

Part of Speech
Parts of Speech of the tweets were extracted using spaCy, especially the pronouns which could be used for identifying an individual *(SubTask_B).*

Others
Other features like tweet polarity (positive, negative or neural score), of hash-tags, of user tags were also used.

4.2 Model Building

Required pre-processing steps like stop-word removal (high and low frequency), stemming, case correction were done. Post which, various features as mentioned above were generated.

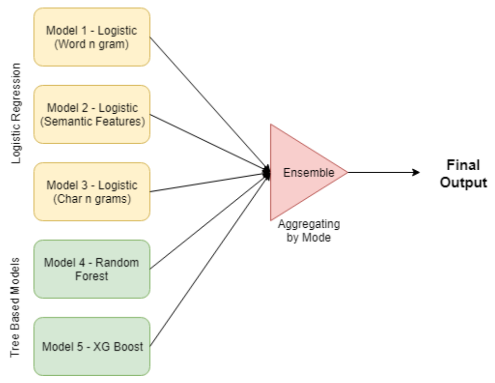


Figure 1: Final Model Architecture

An Ensemble Model was then built by aggregating the results of 5 different models with varying feature set provided as input to each model.

Logistic Regression

Three basic logistic regression models with L2 regularization were developed:

Model 1 was built with Bag of Words (1 - 4 grams) which amounted to 107,445 number of features

Model 2 was built with Tweet Polarity, Word Embedding, Cuss Word Count, and Cuss Word Position

Model 3 was built with character - 4 grams. Various tests with cross validation were performed to arrive at this result with a final count of 100,122 features

Tree Based Models

Model 4 was built using Random forest with Bag of Words (1,2,3,4 grams) containing a total feature size of 107,237.

Model 5 using XG Boost, with Bag of Words (1,2,3,4 grams) containing a feature size of 107,237.

The combined architecture looked as follows,

Vote count was made to arrive at the final decision using the outputs from each of these models

5 Results

5.1 Results - SubTask_A

Model results with respect to the validation data set (part of the training sample) are discussed here. Results of the validation data set with respect to the 80-20 split are shown,

Ensemble Model was able to perform with an accuracy gain of 1.5% with respect to the best individual model *(Model 1)*

Similarly, the results for the model using the given data with an addition of training data provided by TRAC are,

Model	*Accuracy*	*F1 (macro)*
Model1 - Logistic (BOW)	0.78	0.73
Model 2- Logistic (Semantic features)	0.77	0.70
Model 3 - Logistic (Char n gram)	0.76	0.71
Model 3 - Random Forest (BOW)	0.77	0.70
Model 4 - XG Boost (BOW)	0.77	0.71
Ensemble Model	**0.80**	**0.74**

Table 1: Results for SubTask_A without additional data

Model (Additional Data)	*Accuracy*	*F1 (macro)*
Model1 - Logistic (BOW)	0.73	0.72
Model 2- Logistic (Semantic features)	0.70	0.69
Model 3 - Logistic (Char n gram)	0.72	0.71
Model 3 - Random Forest (BOW)	0.71	0.69
Model 4 - XG Boost (BOW)	0.71	0.71
Ensemble Model	**0.74**	**0.73**

Table 2: Results for SubTask_A with additional data

Comparing the results, it can be seen that addition of data in fact reduces the model accuracy.

Features Analysis- Logistic Regression

For better intuitive understanding, top features from logistic regression model trained without additional data were extracted to understand what words constitutes a tweet to be offensive,

Variable	**Coefficient**
stupid	1.798
sucks	1.513
Cuss_word	1.453
crap	1.415
clown	1.274
idiots	1.274
bitch	1.272
sex	1.231

Table 3: Coefficients with higher values

It is clear that words like stupid, sucks, crap and idiots increases the probability of a tweet to be offensive. However, it has been identified that some non-offensive tweets are mis-classified as offensive just because of the presence of such words.

Looking at the coefficients with least weights, it is seen that although the above words have a mild negative connotation, majority of their use-cases are not in an offensive setting which makes a tweet with these to have higher probability of non-offensive class.

Validating the results using actual test data:

Variable	Coefficient
bad	-1.57
mean	-1.08
woman	-1.02
brexit	-0.87
hell	-0.84
fact	-0.76
holy shit	-0.72
pissed	-0.70

Table 4: Coefficients with lower values

The results using ensemble model were submitted to the competition and compared against the actual test data. The table shows the baseline results and the model's performance

System	Accuracy	F1 (macro)
All NOT baseline	0.7209	0.4189
All OFF baseline	0.2790	0.2182
Ensemble - No additiona data	**0.8314**	**0.7565**
Ensemble with additional data	0.8093	0.7433

Table 5: SubTask_A result on actual test data

It was surprising to see that by training the model with additional data, model's accuracy decreased by 3%. This can be mainly attributed to the difference in data sources and difference in response variable definition.

The results between the models trained with and without additional data are to be compared to see the difference between them. Looking at the tweets tagged as Offensive by Model without additional data, but as non-offensive by the other : *@USER Zuckerberg lies.*, *SerenaWilliams is so full of herself...she is just as painful to watch as to listen to...*, and *"50 Cent Calls Out Joe Budden's Bullshit"" On Instagram URL URL.* Looking at the tags, it can be hypothesized that words like 'lies','painful' and 'bullshit' which had very high positive score (offensie), got reduced because of the additional data where these words were not used in an offensive setting. Difference in usage of such words is the reason behind reduction in prediction accuracy for new tweets.

Confusion Matrix

The primary problem is seen with predicting the offensive tweets, where almost half of them are were predicted incorrectly, while a majority of the non-offensive tweets are predicted correctly.

Error Analysis

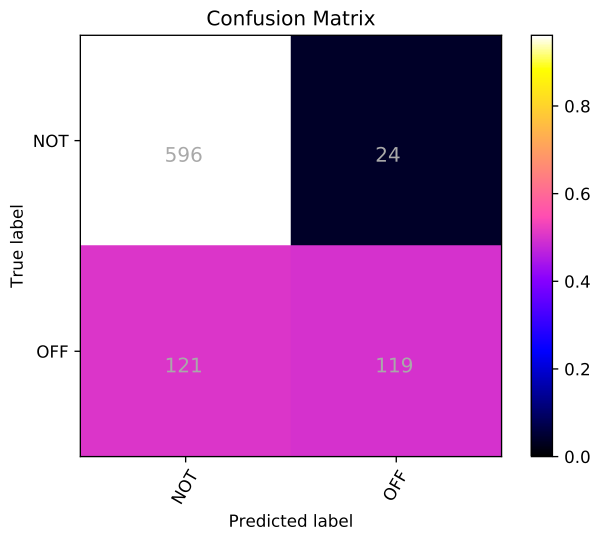


Figure 2: SubTask_A, Ensemble - No additional data

Analyzing the two cases of mis-classification,

i) Offensive tweet tagged non-offensive - While a few of the tweets target an individual or a group, many of them seem to be ambiguous like - *"@USER @USER @USER @USER Kick the absolute shite out of the car", "@USER @USER @USER @USER Yes. Yes he is!"*, and *"Shits about to Hit the Fan. MAGA URL"*

ii) Non offensive being tagged as offensive - The most common reason is the presence of cuss word in a non-offensive sense. Examples are - *Am I a dickhead ???? Probably yes,@USER I've already listened to it like 5 times it's so fucking well made* More features related to the sequence of the sentence, and dependency parsing might help in understanding the syntactic structure

5.2 Results - SubTask_B

The model architecture remained similar to the earlier SubTask, except that Model 2 was trained with additional features like **Parts of Speech** to help detect the target.

The cross-validation results obtained are,

Model (SubTask_B)	*Accuracy*	*F1 (macro)*
Model1 - Logistic (BOW)	0.87	0.46
Model 2- Logistic (Semantic features)	0.87	0.46
Model 3 - Logistic (Char n gram)	0.87	0.45
Model 3 - Random Forest (BOW)	0.86	0.46
Model 4 - XG Boost (BOW)	0.87	0.46
Ensemble Model	**0.88**	**0.47**

Table 6: Results for SubTask_B with no additional data

It is clear that the ensemble model performs bet-

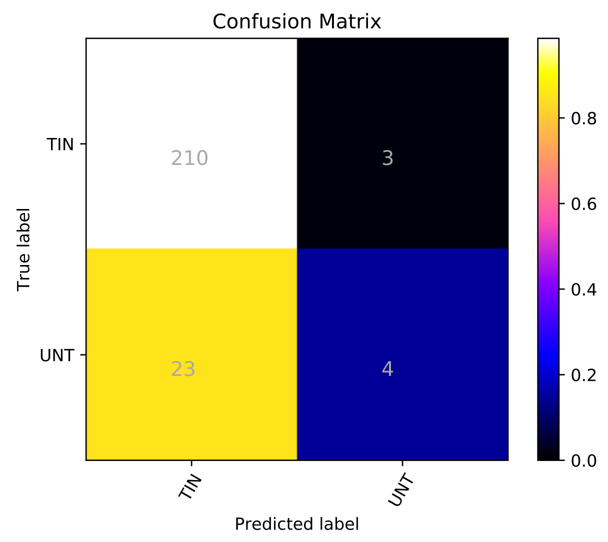


Figure 3: SubTask_B,Confusion matrix for final model

ter than all the other individual models. Now, assessing the performance on actual test data,

System	Accuracy	F1 (macro)
All TIN baseline	0.8875	0.4702
All UNT baseline	0.1125	0.1011
Ensemble Model	0.8917	0.5885

Table 7: SubTask_B result on actual test data

Confusion Matrix

Looking at the confusion matrix in Figure 3, the main issue is seen with predicting a targeted tweet, where almost all of them are predicted incorrectly. This can be ascribed to insufficient number of features that would identify an individual or organization.

6 Conclusion

The proposed ensemble model leverages the best of each of the individual models, where each of the models was experimented with a varying set of features. Some features like word and character n-grams, tweet polarity, cuss word count were more helpful in capturing offensive tweets. The performance for SubTask_B is not appreciable because less number of features related to identifying a target were used. With our best scores for SubTask_A, we were placed 36th out of 103 participants, and were placed 42nd out of 75 submissions in the SemEval-2019 competition. The top team achieved a F1(macro) score of 0.829 in SubTask_A, while we obtained 0.756. Similarly, for SubTask_B the top team had a F1(macro) score of 0.755, while we obtained 0.588. Features like identifying the presence of a person's name using nltk libraries, and the presence of an individual or an organization using Named Entity Recognition with spaCy is highly recommended for further studies.

Moreover, there was a pattern associated with the coefficients having least weights - most of them had a slightly negative connotation. This can justified because of the overall theme of tweets used for training, as most of them were inclined towards *politics*. This lead to overall less number of 'positive words'. Having **more training examples** especially with a range of tweet polarity from more positive to more negative would help in building better models. As seen from our results, an additional data set should be in the same space (Twitter data) to avoid worse performance.

Talking about the offensive tweet categories, there are four types - explicit, implicit, targeted at individuals and groups. The techniques mentioned in this paper using a variety of feature engineering tries to capture most of these categories. However, advanced syntactic features should be introduced to capture patterns like *"Pronoun-Verb-Cuss_word"*. Especially, if a exploratory data analysis is performed on analyzing the patterns of Parts of Speech in offensive tweets, it would help in building additional useful variables.

The main goal of this paper is to show that simpler models which have understandable features can produce good results. More complex methods like introducing polynomial or intricate features, deep learning models using Recurrent Neural Network are other approaches for potentially better accuracy but at the risk of losing interpretability. In conclusion, it is believed that with additional robust features as discussed earlier, the current ensemble machine learning model's performance might increase. Moreover, such features would also be really helpful in interpreting why a tweet could be offensive, which will help in taking necessary actions and remedial measures for social media companies.

References

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 467–472.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics*, pages 1–10, Valencia, Spain.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

YNU-HPCC at SemEval-2019 Task 6: Identifying and Categorising Offensive Language on Twitter

Chengjin Zhou, Jin Wang and **Xuejie Zhang**
School of Information Science and Engineering
Yunnan University
Kunming, P.R. China
`Contact: xjzhang@ynu.edu.cn`

Abstract

This document describes the submission of team YNU-HPCC to SemEval-2019 for three Sub-tasks of Task 6: Sub-task A, Sub-task B, and Sub-task C. We have submitted four systems to identify and categorise offensive language. The first subsystem is an attention-based 2-layer bidirectional long short-term memory (BiLSTM). The second subsystem is a voting ensemble of four different deep learning architectures. The third subsystem is a stacking ensemble of four different deep learning architectures. Finally, the fourth subsystem is a bidirectional encoder representations from transformers (BERT) model. Among our models, in Sub-task A, our first subsystem performed the best, ranking 16th among 103 teams; in Sub-task B, the second subsystem performed the best, ranking 12th among 75 teams; in Sub-task C, the fourth subsystem performed best, ranking 4th among 65 teams.

1 Introduction

Identifying offensive language (Zampieri et al., 2019b) on Twitter is a particularly challenging task because of the informal and creative writing style, with the improper use of grammar, figurative language, misspellings and slang, etc. In previous attempts of the task, OffensEval was generally tackled using hand-crafted features and/or sentiment lexicons by feeding them to classifiers such as Support Vector Machines (SVM). These approaches require a laborious feature-engineering process, which may also need domain-specific knowledge, usually resulting in both redundant and missing features. However, in recent years, artificial neural networks for feature learning have achieved good results in this field (Christos Baziotis, 2017).

SemEval-2019 Task 6 consists of three Sub-tasks (Symeon Symeonidis, 2017):

- Sub-task A: Offensive language identification;
- Sub-task B: Automatic categorisation of offense types;
- Sub-task C: Offense target identification.

In this document, we present four systems that competed at SemEval-2019 Task 6 (Zampieri et al., 2019b). The first model is a 2-layer BiLSTM, equipped with an attention mechanism. The second is voting scheme that combines a 2-layer BiLSTM, Capsule Network, 2-layer bidirectional gated recurrent unit (BiGRU), and the first model. The third model is a stacking scheme that combines a 2-layer BiLSTM, Capsule Network, 2-layer bidirectional gated recurrent unit (BiGRU), and the first model. In addition, the above three models, for the word representation, we have used the glove vector. The fourth model is BERT-BASE (Jacob Devlin, 2018), which was released last year by Google AI Language.

The remainder of this document is organised as follows. The related work is described in Section 2. Section 3 reports our methodology and data. Section 4 reports our result. The conclusions are summarised in Section 5.

2 Related Work

In recent years, with the rapid development of social media, the use of aggressive and offensive language as well as hate speech has gradually increased. To tackle this problematic behaviour, one of the most common strategies is to train systems capable of recognising them and either deleting them or setting them aside for human moderation.

Aggression can be divided into three categories: overt aggression, covert aggression, and non-aggression (Kumar et al., 2018). Last year, in a shared task, several participants used deep neural networks and traditional machine learning methods for aggression identification. The best performing systems in this competition used deep-learning approaches based on convolutional neural networks (CNN), BiLSTM, and long short-term memory (LSTM). Offensive Language is commonly defined as hurtful, derogatory or obscene comments made from one person to another. Currently, there is an increasing amount of such language online. Manually monitoring these posts would incur significant costs (Mathur et al., 2018). Therefore, the automatic identification of suspicious posts has emerges as a trend. In recent years, many researchers have studied the use of deep-learning and traditional machine learning methods for this purpose. Their results indicate that, although several deep-learning approaches produce good scores, traditional supervised classifiers can produce similar scores. Word embeddings, character n-grams and lexicons of offensive words are popular features, but all three components are not necessary for a robust system. Ensemble methods mostly help (Wiegand et al., 2018). Many previous studies still tend to equate offensive language and hate speech. However, through this method, we may erroneously classify many people as hate speakers by failing to differentiate between commonplace offensive language and genuine hate speech (Davidson et al., 2017; Fortuna and Nunes, 2018). In recent years, the recognition of hate speech has mainly focused on deep-learning methods, such as CNNs (Gambäck and Sikdar, 2017) and Convolution-GRU (Zhang et al., 2018).

3 Methodology and Data

3.1 Data

The datasets contain data from Twitter and were provided by the organisers. For Sub-task A, Sub-task B, and Sub-task C, the available datasets (Zampieri et al., 2019a) comprised all the training and testing data. In addition, because the organisers did not provide development, we decided to split 0.2 from the training as development. Table 1 shows the data provided by the organisers.

As shown in Table 1, there the data of the three Sub-tasks shows a significant imbalance.

A	B	C	Train	Test	Total
OFF	**TIN**	**IND**	2,407	100	2,507
OFF	**TIN**	**OTH**	395	35	430
OFF	**TIN**	**GRP**	1,074	78	1,152
OFF	**UNT**	—	524	27	551
NOT	—	—	8,840	620	9,460
ALL			13,240	860	14,100

Table 1: Distribution of label combinations in the data

3.2 Preprocessing

Initially, we received the training and testing data that had been preprocessed by the organisers. Subsequently, on this basis, we preprocessed the training and testing data again and finally applied it to a neural network. For preprocessing, we removed and replaced strings from the tweets that did not show any sentiments, irregularities, or abbreviations. We also removed duplicates and Unicode strings. These were implemented as follows:

- Removing consecutive duplicates while retaining one item: we found that some instances of text were duplicates, e.g. ”????” $\rightarrow$ ”?”.

- Replacing the emojis on Twitter with the corresponding English definition and replacing abbreviations: There were several emojis in the data conveying different emotions. In addition, the abbreviations in the data also restrict the corresponding emotional categories, e.g. ”don’t” $\rightarrow$ ”do not”.

- Replacing irregular words: we found that there were many irregular words in the data, e.g. ”bro” $\rightarrow$ ”brother”.

- Removing some punctuation: preliminary experiments showed better results when we removed some punctuation; however, we detected emotive punctuation signs such as ”!” and ”?” and retained them.

- Converting lowercase: the final tweets were converted to lowercase (after detecting words that had all of their character capitalised, which were retained).

- Using Stanford toolkit: After comparing the use of the word segmentation in the NLTK and Stanford toolkits, we finally decided to use the Stanford toolkit, because of its better performance.

3.3 System

For SemEval-2019 Task 6, we used five basic models:

- **BiLSTM**: BiLSTM is a combination of forward LSTM (LSTM is an artificial recurrent neural network (RNN) architecture; a common LSTM unit comprises a cell, an input gate, an output gate, and a forget gate.) and backward LSTM. Because BiLSTM can better represent bidirectional semantic dependencies, it is often used to model contextual information in natural language processing. In the three Sub-tasks, after several trial comparisons and time factors, we finally selected a 2-layer BiLSTM. In addition, the parameters of our model were chosen to maximise development performance: in Sub-task A, we initialised the hidden dimension, recurrent dropout, and batch size as 120, 0.25, and 128, respectively; in Sub-task B, we initialised the hidden dimension, recurrent dropout, and batch size as 120, 0.25, and 100, respectively; and in Sub-task C, we initialised the hidden dimension, recurrent dropout, and batch size as 140, 0.35, and 64, respectively.

- **BiGRU**: similarly, BiGRU is a combination of forward GRU (GRU, a variant of LSTM, has a simpler structure than LSTM and works well; there are only two gates in the GRU model, namely the update gate and the reset gate) and backward GRU. For the three Sub-tasks, we used a 2-layer BiGRU. The parameters of our model were chosen to maximise development performance: in Sub-tasks A and B, we initialised the hidden dimension, recurrent dropout, and batch size as 120, 0.25, and 100, respectively; in Sub-task C, we initialised the hidden dimension, recurrent dropout, and batch size as 120, 0.25, and 128, respectively.

- **BiLSTM with attention**: For this, an attention layer was added to the 2-layer BiLSTM. In BiLSTM, we used the output vector of the last time sequence as the feature vector and then performed softmax classification. The attention layer is used to first calculate the weight of each time sequence, then take the weighted sum of all the time sequence vectors as feature vectors, and finally perform softmax classification. Similar to the previous models, the parameters of our model were as follows: in Sub-tasks A and B, we initialised the hidden dimension, recurrent dropout, and batch size as 120, 0.25, and 256, respectively; in Sub-task C, we initialised the hidden dimension, recurrent dropout, and batch size as 180, 0.3, and 128, respectively.

- **Capsule Network**: In the deep-learning model, the spatial patterns are summarised at the lower level, thus helping represent the concept of higher layers. For example, when a CNN models spatial information, it needs to copy the feature detector, which reduces the efficiency of the model. However, spatially insensitive methods are inevitably limited by rich text structures (such as the preservation of word location information, semantic information, and grammatical structure), which are difficult to encode effectively and lack text expression ability. Hinton et al. (Sara Sabour, 2017) proposed a Capsule Network, which replaces a single neuron node of a traditional neural network with a neuron vector and trains this new neural network through dynamic routing, effectively improving the shortcomings of the above two methods. The parameters of our model were as follows: in Sub-tasks A and B, we initialised the hidden dimension, batch size, and routing as 64, 120, and 15, respectively; in Sub-task C, we initialised the hidden dimension, batch size, and routing as 64, 140, and 15, respectively.

- **BERT**: The BERT model is a language model proposed by Google based on a bidirectional transformer. It is quite different from ELMo (Peters et al., 2018). In existing pre-training models (including word2vec and ELMo), word vectors are generated. This type of pre-training model belongs to domain transfer. The GPT (Karthik Narasimhan and Sutskever, 2018), BERT, etc. proposed in recent years are all examples of model migration. Furthermore, the BERT model combines the pre-training model with the downstream task model. In other words, it is still utilised when performing downstream tasks, and text classification tasks are naturally supported. The model does not need to be modified when performing text classification tasks. The BERT model has two versions on the English datasets, namely Base and Large, and we used the Base version. The parameters of our model were as follows: transformer blocks (L) was set as 12, hidden size (H) as 768, number of self-attention heads (A) as 12, total parameters as 110M, train batch size as 32, predict batch size as 8, and learning rate as 0.00002.

For the four models of BiLSTM, BiGRU, BiLSTM with attention, and Capsule Network, first, the processed Twitter text was converted into a word vector matrix. Then the word vector matrix was processed by the embedded layer. Subsequently, the word vector matrix was converted to a computable vector matrix. Finally, the four models could utilise the vector matrix for training and prediction.

3.4 K-Fold Cross-Validation

We know from Section 3.1 that data imbalance exists in the public datasets published by the organisers. This would lead to unstable or inaccurate experimental results. To manage this problem, we used k-fold (k = 5) cross-validation: the training sample was randomly partitioned into 5 equal sized subsamples. Of the 5 subsamples, a single subsample was retained as validation data to test the model, and the remaining 4 subsamples were used as training data.

4 Results

4.1 Task A

Sub-task A includes 13240 training instances and 860 testing instances, as well as OFF and NOT labels. We used four models for predictions on the testing sets. These four models were BERT (system ID: 528280), voting (system ID: 528117), stacking (system ID: 528015), and BiLSTM with attention (system ID: 528232). In the voting model, we performed soft voting ensemble on four basic models: BiLSTM, BiGRU, BiLSTM with attention, and Capsule Network. In the stacking model, we performed stacking ensemble on four basic models: BiLSTM, BiGRU, BiLSTM with attention, and Capsule Network. Our team results according to those provided by the task organisers are shown in Table 2. Among the results of the four models submitted by our team, the BiLSTM with attention model performed the best, and its F1 (macro) was 0.7877. The accuracy was 0.843, ranking 16th among all participants. In addition, from the confusion matrix in Figure 1, it is observed that when the classifier predicts two classes of labels, namely NOT and OFF, it is more specific to the NOT label, and the precision for the NOT label is higher than that for the OFF label.

System ID	F1 (macro)	Accuracy
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
528015	0.7258	0.7872
528117	0.7817	0.836
528280	0.7667	0.8174
528232	0.7877	0.843

Table 2: Results for Sub-task A

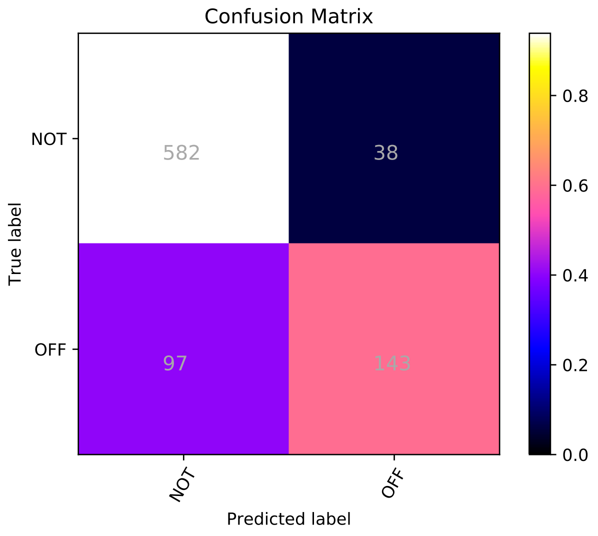


Figure 1: Sub-task A, YNU-HPCC CodaLab 528232

4.2 Task B

Sub-task B continues on the OFF label of Sub-task A. It includes 4400 training instances and 240 testing instances, as well as TIN and UNT labels. We used BERT (system ID: 533313), voting (system ID: 533291), and BiLSTM with attention (system ID: 533311) for predictions on the testing sets. The results of our team according to those provided by the task organisers are shown in Table 3. Among the results of the three models submitted by our team, the voting model performed best; its F1 (macro) was 0.6811, its accuracy was 0.8625, and it ranked 12th among all participants. Similar to the previous Sub-task, the confusion matrix in Figure 2 indicates that, for the TIN and UNT labels, the classifier is more sensitive to TIN labels. In terms of precision, the value for the TIN label is also higher than that for the UNT label.

System ID	F1 (macro)	Accuracy
All TIN baseline	0.4702	0.8875
All UNT baseline	0.1011	0.1125
533291	0.6811	0.8625
533311	0.6248	0.7833
533313	0.6530	0.8375

Table 3: Results for Sub-task B

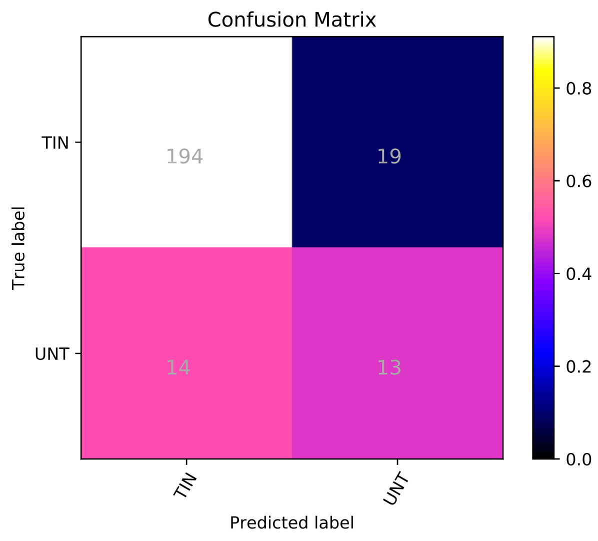


Figure 2: Sub-task B, YNU-HPCC CodaLab 533291

4.3 Task C

Sub-task C continues on the TIN label of Sub-task B. It includes 3876 training instances and 213 testing instances, as well as IND, OTH, and GRP labels. We used BERT (system ID: 536705) and voting (system ID: 537472) for predictions on the testing sets. The results of our team according to those provided by the task organisers are shown in Table 4. Among the results of the two models submitted by our team, the BERT model performed the best; its F1 (macro) was 0.6212, its accuracy was 0.7089, and it ranked 4th among all participants. Additionally, as shown in Figure 3, among the IND, OTH, and GRP labels, the highest recall and precision are for the IND labels, and the lowest are for the OTH labels.

For the three Sub-tasks, misclassifications of the classifier are likely due to data imbalance.

System ID	F1 (macro)	Accuracy
All GRP baseline	0.1787	0.3662
All IND baseline	0.2130	0.4695
All OTH baseline	0.0941	0.1643
536705	0.6212	0.7089
537472	0.5377	0.6667

Table 4: Results for Sub-task C

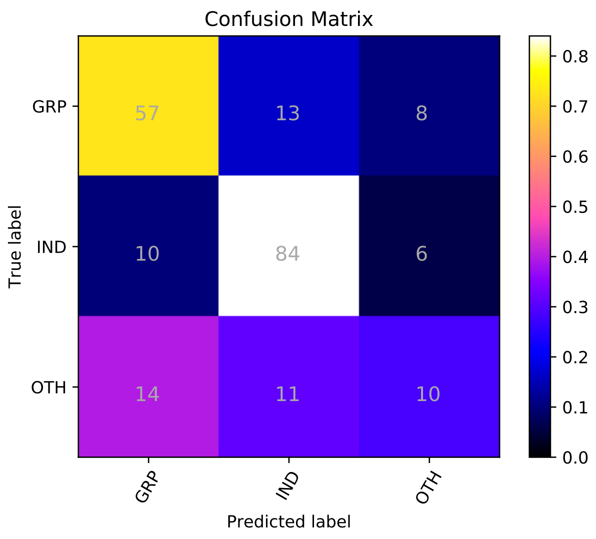


Figure 3: Sub-task C, YNU-HPCC CodaLab 536705

5 Conclusion

Identifying and categorising offensive language is a task that is drawing increasing attention. In this document, we described our four models submitted for Task 6 of the SemEval-2019 Workshop, which involved identifying and categorising offensive language on Twitter. These four models comprise not only traditional neural network models but also popular language models. Our model exhibited good performance in terms of the experimental results. In the three Sub-tasks, there appears to be significant room for improvement compared to the top-ranked participating systems. Therefore, in future work, we will focus on using more word embedding methods and managing data imbalance issues.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants No.61702443 and No.61762091, and in part by Educational Commission of Yunnan Province of China under Grant No.2017ZZX030. The authors would like to thank the anonymous reviewers and the area chairs for their constructive comments.

References

Christos Doulkeridis Christos Baziotis, Nikos Pelekis. 2017. DataStories at SemEval-2017 Task 4:Deep L-STM with Attention for Message-level and Topic-based Sentiment Analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, pages 747–754, Vancouver, Canada.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Kenton Lee Kristina Toutanova Jacob Devlin, Ming-Wei Chang. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs.CL]*.

Tim Salimans Karthik Narasimhan, Alec Radford and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training. *OpenAI*.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Puneet Mathur, Rajiv Shah, Ramit Sawhney, and Debanjan Mahata. 2018. Detecting offensive tweets in hindi-english code-switched language. In *Proceedings of the 6th International Workshop on Natural Language Processing for Social Media*, pages 18–26.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Geoffrey E Hinton Sara Sabour, Nicholas Frosst. 2017. Dynamic Routing Between Capsules. *arXiv:1710.09829 [cs.CV]*.

John Kordonis Avi Arampatzis Symeon Symeonidis, Dimitrios Effrosynidis. 2017. DUTH at SemEval-2017 Task 4: A Voting Classification Approach for Twitter Sentiment Analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, pages 704–708, Vancouver, Canada.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

YNUWB at SemEval-2019 Task 6: K-max pooling CNN with average meta-embedding for identifying offensive language

Bin Wang, Xiaobing Zhou[*] **,Xuejie Zhang**
School of Information Science and Engineering
Yunnan University, Yunnan, P.R. China
[*]Corresponding author`zhouxb@ynu.edu.cn`

Abstract

This paper describes the system submitted to SemEval 2019 Task 6: OffensEval 2019. The task aims to identify and categorize offensive language in social media, we only participate in Sub-task A, which aims to identify offensive language. In order to address this task, we propose a system based on a K-max pooling convolutional neural network model, and use an argument for averaging as a valid meta-embedding technique to get a meta-embedding. Finally, we use a cyclic learning rate policy to improve model performance. Our model achieves a Macro F1-score of 0.802 (ranked 9/103) in the Sub-task A.

1 Introduction

In the past ten years, with the popularity of the Internet, social media platforms such as facebook and twitter have gradually become important tools for people's daily communication, and users can publish their own content on these platforms. As the number of people interacting on social media platforms increases, online aggression language behavior also grows, and now it has become a major source of social conflict.

Semeval 2019 Task 6 is proposed for identifying online offensive languages (Zampieri et al., 2019b). Its goal is to use computational methods to identify offense, aggression and hate speech in user-generated content on online social media platforms. We can prevent abuse of offensive language by using this approach in social media platforms. This task gives us some data from the social media platform, and classifies the content through computational analysis.

In this competition, we only participate in Subtask A: identification of offensive language. For this task, we use a deep learning method to build a K-max pooling convolutional neural network model which uses convolutional neural networks of different filters to extract features and preserves k largest eigenvalues during the pooling phase. We use two different pre-training vector models (fastText and Glove) to obtain a more accurate meta-embedding with a simple averaging technique (Coates and Bollegala, 2018). In addition, we have adopted a Cyclic Learning Rate (CLR) strategy (Smith, 2017), which avoids the process to find the optimal learning rate, and the learning rate varies within a reasonable interval rather than monotonically. Unlike the Adative Learning Rate, the CLR does not require additional calculations.

The rest of the paper is structured as follows: In section 2, we describe some of the relevant research work. In section 3, we describe the task data and how to build the model. In section 4, we describe the experimental results.

2 Related Work

In recent years, offensive language has prevailed in social media, and people are increasingly interested in identifying offensive speech, especially on social media platforms. This topic has attracted the attention of a large number of researchers in industry and academia. The field of Hate Speech Automatic Detection in the text has unquestionable social impact potential, especially in online communities and digital media platforms (Fortuna and Nunes, 2018). In this section, we will review some of the studies and briefly discuss their findings.

Dinakar et al. decomposed the overall detection problem into detection of sensitive topics, incorporated itself into the text classification subquestion, and solved the problem of text cyberbullying detection by constructing a separate topic-sensitive classifier(Dinakar et al., 2011). Burnap et al. used probabilities, based on the combination of rules and space-based classifiers and voting element classifiers to predict the possible spread of network hatred in Twitter data samples (Burnap and Williams, 2015). Kwok et al. used supervised machine learning methods to obtain tagged data from different Twitter accounts in an inexpensive way, and to learn the binary classifiers of the "racist" and "nonracist" tags (Kwok and Wang, 2013). Gambäck et al. built a convolutional neural network model based on word2vec embedding(Gambäck and Sikdar, 2017). And a new method for deep neural networks based on convolution and gated recursive networks was proposed by Zhang et al. (Zhang et al., 2018). In the field of research on hate speech, originally Xu et al. introduced the social study of bullying and formulated it as NLP tasks(Xu et al., 2012), then Ross et al. suggested that the existence of hate speech should not be considered as a binary yes or no decision, and the evaluator needs a more detailed commentary(Ross et al., 2016), and now ElSherief et al. believed that it is necessary to further deepen the understanding of online hate language to determine whether the target is individual or group (ElSherief et al., 2018).

3 Methodology and Data

3.1 Data description

In this task, we only use the official training data set for training and trial data set to verify. The official data provided by OLID is mainly from Twitter (Zampieri et al., 2019a). In Sub-task A, the purpose is to distinguish whether the tweet is offensive, so the data is divided into two categories: Not Offensive (NOT): Posts that do not contain offensive or defamatory; and Offensive (OFF): This category includes insults, threats, and posts that contain defamatory or cursed words. The training data set has a total of 13240 tweets, in which there are 8840 of NOT and 4400 of OFF, the ratio is about 2:1. The data is slightly unbalanced, but we have not dealt with the data imbalance problem.

3.2 K-max pooling CNN model

Our network architecture is shown in Figure 1. It is a variant of the CNN model structure proposed by Yoon Kim (Kim, 2014). Next we explain the details of our system.

- **Input layer:** This layer mainly inputs all the preprocessed text data into the model.

- **Embedding layer:** Converting text into word embeddings represents each word of the text with a d dimensional vector by using a pre-trained word vector model.

- **Convolutional layer:** In this layer, the obtained word vectors are subjected to convolution operations to obtain multiple feature maps. The specific operation is: a sentence contains L words, each of which has a dimension of d after the embedding layer, and forms a $L * D$ sentence representation by splicing L words. There are several convolution kernels in the convolutional layer, the size of which is $N * d$, and N is the filter window size. The convolution operation is to apply a convolution kernel to create a new feature in a matrix that is spliced by words. Its formula is as follows:

$$C_l = f(w * x_{(l:l+N-1)} + b) \qquad (1)$$

where l represents the lth word, c_l is the feature, w is the convolution kernel, b is the bias term, and f is a nonlinear function. After the convolution operation of the whole sentence, a feature map is obtained, which is a vector of size $L + N - 1$.

- **Pooling layer:** The main function of this layer is to perform dimensionality reduction on the features of filter to form the final feature with a K-max pooling operation, which takes the value of the scores in Top K among all the feature values, and retains the original order of these feature values. Obviously, K-max Pooling can express the same type of feature multiple times, that is, it can express the intensity of a certain type of feature; In addition, because the relative order of these Top K eigenvalues is preserved, it should be said that it retains part of the position information. However, this location information is only the relative order between features, not absolute location information. For example: "I think the scenery in this place is not bad, but there are too many people." Although the first half reflects the positive emotions, the global text expresses the negative emotions, and K-max pooling can capture such information.

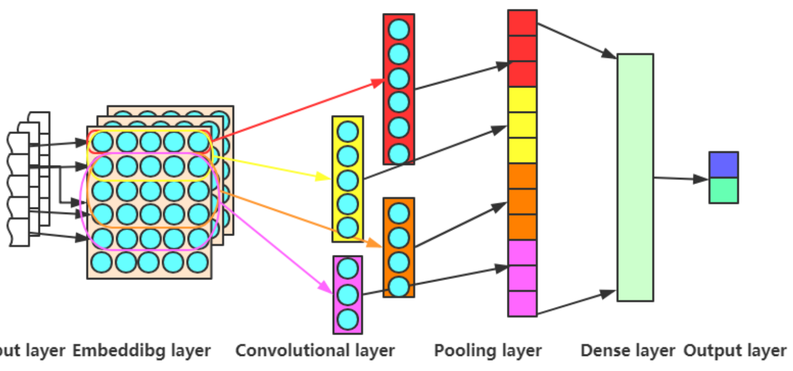


Figure 1: The architecture of K-max pooling CNN model

- **Fully connected layer:** In this model architecture, there are two layers of fully connected layers. The first layer receives the feature vectors obtained by the pooling layer, and the last layer is used for classification and prediction.

3.3 Word embedding

We use two different pre-trained word embeddings, fastText and Glove. FastText is provided by Mikolov et al. (Mikolov et al., 2018), it is a 2 million word vector trained using subword information on Common Crawl with 600B tokens, and its dimension is 300. Glove is provided by Jeffrey Pennington et al. (Pennington et al., 2014), it is a 2.2 million word vector trained using subword information on Common Crawl with 840B tokens, and its dimension is also 300.

We use a mathematical mean of the word vectors by fastText and Glove to produce a high performance word vector. Since the principles between fastText and Glove are different, the word vector representation of the same word is also slightly different, and the average embedding set retains semantic information through preservation of the relative distances between words (Coates and Bollegala, 2018).

3.4 CLR

In this article, we use the cyclic learning rate strategy provided by Leslie N . Smith (Smith, 2017), which is a new method of setting the global learning rate. The advantage is that it avoids a lot of experiments to find the optimal learning rate and can be faster. We set the base learning rate and the maximum learning rate so that the learning rate fluctuates cyclically within this interval. The wave method we use is exp_range, which is a triangle loop that scales the loop magnitude by a factor while keeping the initial learning rate constant.

4 Experiment and results

4.1 Data preprocessing

Text from tweets are inherently noisy. Tweets are processed using tweettokenize tool. Cleaning the text before further processing helps to generate better features and semantics. We perform the following preprocessing steps.

- The “#” symbol is removed and the word itself is retained for hashtags.
- All of “@user” is replaced with username. Username mentions, i.e. words starting with “@”, generally provide no information in terms of sentiment. Hence such terms are removed completely from the tweet.
- Repeated full stops, question marks and exclamation marks are replaced with a single instance with a special token ”repeat” added.
- All contractions are split into two tokens(e.g.: “it’s” is changed to “it” and “is”).
- Emoticons (for example, ‘:(’, ‘:)’, ‘:P’ and emoji etc) are replaced with their own meanings by emotion lexicons.
- Lemmatization, restoring language vocabulary to general form (can express complete semantics) by WordNetLemmatizer.
- Tokens are converted to lower case.

4.2 Experiment setting

We use 4-fold cross validation on the training data, because in this experiment we experimented with cross-validation of different k values, and found that the 4-fold cross-validation effect is the best. In addition the batch size is to 512 and the epoch to 20. In our model, the dimension of embeding is 300. Between the embedding layer and the convolution layer we add the SpatialDropout1D layer with a value of 0.2. In the convolution layer, we set up four convolution kernels of different window sizes, which are 1, 2, 3 and 4, the number of filters is 180, the kernel initializer is $normal$, the activation function is $relu$; in the pooling layer we set the k value to 3. Before the fully connected layer, we add a dropout layer, and the rate is 0.6. The activation function of the final output layer is $sigmoid$ for binary classification. The loss function of this model is $binary\ crossentropy$, and the optimizer is $adam$.

For the cyclical learning rate, we set the base learning rate to 0.001, the maximum learning rate to 0.002, the step size to 300, and the scaling factor gamma to 0.99994.

4.3 Result analysis

This Sub-task A is to evaluate the classification system by calculating the marco F1 score. According to the official ranking of Sub-task A, our model has a marco F1 score of 0.8024, ranked 9th, and our result is much higher than the official baseline. The results of the official baseline and our model are shown in Table 1.

System	F1 (macro)	Accuracy
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
Our model	**0.8024**	**0.8453**

Table 1: Results for Sub-task A

The confusion matrix of our model prediction results in Sub-task A is shown in Figure 2. There are 620 NOT tags in the test dataset, and 240 OFF tags. As can be seen from the confusion matrix, our model has a lot of OFF prediction errors into NOT, about 32% of OFF are predicted to be NOT, while only about 9% of NOT are predicted to be OFF.

4.4 Influence of Word Embedding

The effect of the average meta-embedding technique is shown in Table 2. In this table, the results of fastText, Glove, the word vector generated by concatenating fastText and Glove and the vector

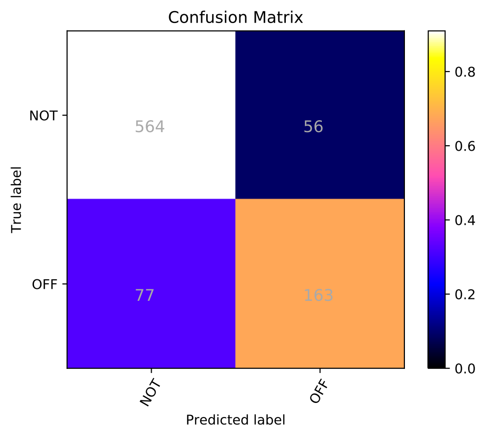


Figure 2: Confusion matrix of K-max pooling CNN model for Sub-task A

generated by the average meta-embedding technique are compared, and the result is obtained in the trial data set. The results show that this average meta-embedding technique can improve the performance of the model.

word vector	F1 (macro)
fastText	0.8138
Glove	0.7895
concatenated	0.8165
average meta-embedding	**0.8242**

Table 2: Influence of word embedding in trial data set for Sub-task A

As can be seen from Table 2, the concatenated can also improve the performance of the model, but increases the dimension of word embeddings to 600. While the meta-embedded dimension will not exceed the maximum dimension existing in the source embedding.

5 Conclusion

In this paper, we present a K-max pooling convolutional neural network model based on average meta-embedding technology for offensive language detection, which relies solely on the data sets provided to generate competitive results. However, the data imbalance problem will affect the performance of the model, which makes the model prediction tend to be biased towards a high amount of data. In the future work, we will strengthen the processing of data imbalance problems, and try to extract some NER features from the data to further improve the performance of the model.

Acknowledgments

This work was supported by the Natural Science Foundations of China under Grant Nos. 61463050, 61702443 and 61762091, and the Project of Innovative Research Team of Yunnan Province under Grant No. 2018HC019.

References

Pete Burnap and Matthew L Williams. 2015. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet*, 7(2):223–242.

Joshua Coates and Danushka Bollegala. 2018. Frustratingly easy meta-embedding computing meta-embeddings by averaging source word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers).*

Karthik Dinakar, Roi Reichart, and Henry Lieberman. 2011. Modeling the detection of textual cyberbullying. In *The Social Mobile Web*, pages 11–17.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257.*

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882.*

Irene Kwok and Yuzhou Wang. 2013. Locate the hate: Detecting Tweets Against Blacks. In *Twenty-Seventh AAAI Conference on Artificial Intelligence.*

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018).*

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Wojatzki. 2016. Measuring the Reliability of Hate Speech Annotations: The Case of the European Refugee Crisis. In *Proceedings of the Workshop on Natural Language Processing for Computer-Mediated Communication (NLP4CMC)*, Bochum, Germany.

Leslie N Smith. 2017. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from bullying traces in social media. In *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 656–666. Association for Computational Linguistics.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL.*

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval).*

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

Zeyad at SemEval-2019 Task 6: That's Offensive! An All-Out Search For An Ensemble To Identify And Categorize Offense in Tweets

Zeyad El-Zanaty
Faculty of Engineering
Alexandria University
zeyadzanaty@gmail.com

Abstract

The objective of this paper is to provide a description for a classification system built for SemEval-2019 Task 6: OffensEval. This system classifies a tweet as either offensive or not offensive (Sub-task A) and further classifies offensive tweets into categories (Sub-tasks B - C). The system consists of two phases; a brute-force grid search to find the best learners amongst a given set and an ensemble of a subset of these best learners. The system achieved an F1-score of 0.728, ranking in subtask A, an F1-score score of 0.616 in subtask B and an F1-score of 0.509 in subtask C.

1 Introduction

In OffensEval we break down offensive content into three sub-tasks taking the type and target of offenses into account. Sub-task A - Offensive language identification; In this sub-task we are interested in the identification of offensive posts and posts containing any form of (untargeted) profanity. In this sub-task there are 2 categories in which the tweet could be classified Not Offensive - This post does not contain offense or profanity. Non-offensive posts do not include any form of offense or profanity. Sub-task B - Automatic categorization of offense types; In this sub-task we are interested in categorizing offenses. Tweets are labeled from one of the following categories Targeted Insult - A post containing an insult or a threat to an individual, group, or others; Untargeted - A post containing non-targeted profanity and swearing. Posts containing general profanity are not targeted but they contain non-acceptable language. On the other hand, insults and threats are targeted at an individual or group. Sub-task C - Offense target identification. Finally, in sub-task C we are interested in the target of offenses. Only posts which are either insults or threats are included in this sub-task. The three categories included in sub-task C are the following: Individual - The target of the offensive post is an individual: a famous person, named individual or an unnamed person interacting in the conversation. Group - The target of the offensive post is a group of people considered as a unity due to the same ethnicity, gender or sexual orientation, political affiliation, religious belief, or something else. Other The target of the offensive post does not belong to any of the previous two categories (e.g. an organization, a situation, an event, or an issue).(Zampieri et al., 2019b)

To work with such complicated tasks our approach is an exhaustive one. We try combinations of many techniques in pre-processing, feature extraction and classification while tuning their hyper-parameters to find the best models with the leading F1-scores. With the gained information an ensemble of the top three models is formed to get the optimum result. Along side to this approach we try a deep-learning method with a simple 1D - CNN consisting of 3 convolutional layers and a softmax layer just to compare results.

2 Related Work

Offensive language on social media hardly remains unnoticed. Contents involving hateful messages vary from hate speech to group-based racism and could target anyone irrespective of their status, identity, location and so forth. Even when it is not materialized into a hate-motivated crime, the damage is done victims are being labeled, marginalised and exposed to negative stereotyping. The overall consequences of online hate can be the dehumanisation of individuals or groups of individuals. The need for proper strategies to tackle hate speech on social media

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 823–828
Minneapolis, Minnesota, USA, June 6–7, 2019. ©2019 Association for Computational Linguistics

is unquestionable. The core focus of the thesis is not to find a solution to the challenge, but rather to identify central problems that have contributed to the formation of the existing reality. To unrave the contributing factors, a holistic analysis of both international human rights principles regarding hate speech and the practical application of those standards is necessary.(Schofield and Davidson, 2017)

There have been many studies and publication on the topic of offensive language and hate speech over the last few years. Examples on such studies include (Davidson et al., 2017), (Malmasi and Zampieri, 2017), (ElSherief et al., 2018), (Gambäck and Sikdar, 2017), (Zhang et al., 2018). Also there have been challenges on how to distinguish profanity from hate-speech presented by (Malmasi and Zampieri, 2018).

3 Methodology and Data

The used dataset in this assignment is the one provided in SemEval-2019 task 6. The dataset has been collected from Twitter. It was retrieved by searching offensive terms that could be present in a tweet. It consists of 14,100 tweets in total. It was annotated using crowdsourcing. The gold labels were assigned taking the agreement of three annotators into consideration. No correction has been carried out on the crowdsourcing annotations. The dataset was presented in two phases; Training data: already labeled tweets used to train the classifiers. Each tweet was provided with a binary classification label and an index. Testing data: unlabeled tweets to test the classifiers against. Zampieri et al. (2019a).

The system is a combination of three essential layers. First, pre-processing which is a necessary step in NLP as textual data could and most likely is not clean, thus will affect further stages and create an incoherent model. Second, feature extraction or vectorization, which translates words to a number or a series of numbers with different weights to represent this word. Finally, classification, features extracted from the previous step is fed into a learner and a model is created that could classify tweets.

For our approach we implemented a heap of pre-processors, vectorizers and classifiers and with the help of brute-froce search ranked all the resulting models according to their F1-scores. All implemented techniques are available in Table 1. For the implementation see: `github.com/zeyadzanaty/offenseval`

Phase	Implemented Techniques
Pre-processing	Stopwords Removal Lemmatization - Stemming
Feature Extraction	TFIDF - Count - Word Embedding
Classification	KNN - Naive Bayes - Decision Trees - SVM -Random Forest Logistic Regression - MLP

Table 1: Multiple techniques implemented in our system.

3.1 Pre-processing

A tweet contains many unwanted data that would take extra computational power and decrease the accuracy of the model. So, noise removal and some normalization techniques must be applied to the corpus in-order to generate more consistent models. **Stopword Removal** is a noise removal method by filtering words that dont have significance in the context of the sentence, without them the semantics of the tweet wont be affected. **Lemmatization** is the process of getting the linguistic root of a word. First, words are part-of-speech tagged , then converted to their roots. **Stemming** is the process of stripping a word of it's prefixes and suffixes using the porter-stemmer algorithm (Porter, 1980).

For this step, a list of all combinations of pre-processing techniques is used. For example it would look something like: [(Stopwords Removal), (Stowords Removal, Lemmatization), (Stopwords Removal, Stemming), (Lemmatization), etc..]

3.2 Feature Extraction

Now that we've got our clean, almost noise-free textual data, we cant simply feed a classification model a bunch of text words, most models only work with numerical data. This is where we convert words to numerical features using

one the methods mentioned below to create our classification-ready data.
We use three word embedding models of embedding dimension 100 (which gave adequate results after experimenting with other dimensions) along side to the standard **TFIDF/Count** models.

- Word2Vec model trained on our dataset.(Mikolov et al., 2013)
- fastText model trained on our dataset.(Joulin et al., 2016)
- Pre-trained GloVe model trained on 2 Billion tweets - 27 Billion tokens - 1.2 million vocabulary.(Pennington et al., 2014)

All three models mentioned are zero-padded with the maximum length of a tweet present in the dataset to resolve the uneven dimensionality issue. A list of all techniques is initialized for later usage in the search for the best model.

3.3 Classification and Tuning

This is where all the previous work comes together for the final phase of the system. Seven models where chosen and tuned using sci-kit learns (Pedregosa et al., 2011) GridSearchCV, which does a cross validation search on a list of hyper-parameters for a given model. The parameters grids that were tested are available in Table 2.

Model	Parameters Grid
KNN	n_neighbours: [1, 3, 5, 7]
Naive Bayes	fit_prior: [True, False]
SVM	C: [0.1,10,100] kernel: [rbf, poly]
Decision Trees	criterion: [gini, entropy]
Random Forest	n_estimators: [10 - 200]
Logistic Regression	penalty : [l2] solver: [sag, lbfgs, newton]
MLP	activation:[tanh, relu] solver: [sgd,adam, lbfgs]

Table 2: Classification models and their corresponding parameters to tune.

Again, a list of classifiers and their parameters grids is initialized to tune them with a 3-fold cross validation.

3.4 All-Out Search

This is the body of all the work. We try every possible combination of pre-processing, vectorization and classification to ensure the output has the best possible F1-score for the given subtask. We start by cleaning the data using a certain combination of pre-processors, then extracting features using one of the vectorizers and finally to complete the pipeline, tune a classifier's hyper-parameters on the resulting data-matrix. And repeat for the next combination.

```
1: procedure SEARCH($preprocessors$, $vectorizers$, $classifiers$)
2:     $models \leftarrow \{\}$
3:     for $prp \in preprocessors$ do
4:         clean-data(prp)
5:         for $vec \in vectorizers$ do
6:             vectorize-data(vec)
7:             for $clf \in classifiers$ do
8:                 $models[clf] \leftarrow tune(clf)$
9:     $sort(models)$
```

The resulting set 'models' is a set of each classifier and a list of parameters and their corresponding preprocessors, vectroizers and F1-scores. The results could be plotted to help visualize the performance of each model seen in Figure 1, which

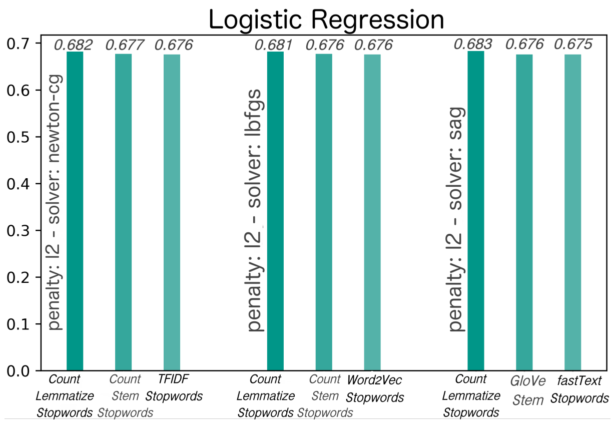


Figure 1: F1-scores of logistic regression model on subtask A, each bar is a model with it's own pre-processing, vectorizer and parameters.

shows the top 3 models for the logistic regression classifier. Each 3 bars represent the hyper-parameters combination and the top 3 combinations of pre-processing and vectorization. The best F1-score (0.683) came from a pre-processing of stopwords removal followed by lemmatization, a count vectroizer and hyper-parameters [penalty: l2, solver: sag]. Following this search, now that

Subtask	Phase	1st	2nd	3rd
A	Pre-processing	Stopwords Removal & Lemmatization	Stopwords Removal & Stemming	Lemmatization
	Vectorization	Count	Count	Count
	Classification	Logistic Regression	Naive Bayes	Random Forest
B	Pre-processing	Lemmatization	Lemmatization	Stopwords-Removal
	Vectorization	TFIDF	TFIDF	GloVe-Embeddings
	Classifiaction	Naive Bayes	Random Forest	MLP
C	Pre-processing	Lemmatization	Stopwords Removal	Stemming
	Vectorization	Count	Count	Count
	Classification	Random Forest	Logistic Regression	Naive Bayes

Table 3: Top 3 models for each subtask, these 3 models will form an ensemble to enhance the performance.

we have the scores of each model, we can model an ensemble of the top 3 models to give us a better overview of the data available in Table 3. And just to add an extra layer we can re-tune the classifier parameters in case of any error that could have appeared in the previous step.

4 Results

We submitted with a couple of models, for subtask A, an ensemble of the three top models mentioned in Table 3, the Random Forest model and a 1-D CNN. The ensemble did its best in subtask A but the Random Forest (RF) model came a very close second. Results can be viewed in Table 4, and confusion matrix Figure 2.

System	F1 (macro)	Accuracy
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
Ensemble	**0.7289**	**0.8151**
Random Forest	0.7143	0.8128
1D-CNN	0.5506	0.6977

Table 4: Results for Sub-task A. The ensemble approach gave the best results.

As for subtask B, the ensemble submission unfortunately failed, but it didn't look good anyway. The best model was as simple Naive Bayes (NB)-TFIDF model which got a very good F1-score of 0.887. Results can be viewed in Table 5, and confusion matrix Figure 3.

System	F1 (macro)	Accuracy
All TIN baseline	0.4702	0.8875
All UNT baseline	0.1011	0.1125
1D-CNN	0.4436	0.5542
Naive Bayes	**0.6161**	**0.8542**

Table 5: Results for Sub-task B. The best performer was a simple TFIDF - Naive Bayes model.

Finally, for subtask C, we chose to let go of the CNN model as it didn't get an acceptable result, and went for the ensemble, which got the best accuracy but came second for F1-scores and the RF model which also got a good accuracy but a poor F1-score, and the best model was a logistic regression-count model with an F1-score of 0.5093. Results can be viewed in Table 6, and confusion matrix Figure 4.

System	F1 (macro)	Accuracy
All GRP baseline	0.1787	0.3662
All IND baseline	0.2130	0.4695
All OTH baseline	0.0941	0.1643
Ensemble	0.4973	0.6479
Random Forest	0.4763	0.6432
Logistic Regression	**0.5093**	**0.6056**

Table 6: Results for Sub-task C. The ensemble got the best accuracy but, LR got a better F1-score.

Looking at these results, we hypothesize that the systems performance can be improved by combining all word embedding features instead of using them individually. It was also remarkable that the for most subtasks a simple Naive Bayes - TFIDF model came close to being the best amongst all others. We also believe better results can be achieved if there was the dataset was more balanced and having more offensive tweets, and if we had sufficient time to perform grammar checking on the tokens and other operations that can reduce noise. The problem of out-of-vocabulary (OOV) words which we unfortunately didn't attempt to solve, could be later be solved by using a character-level embedding model rather than a word embedding one.

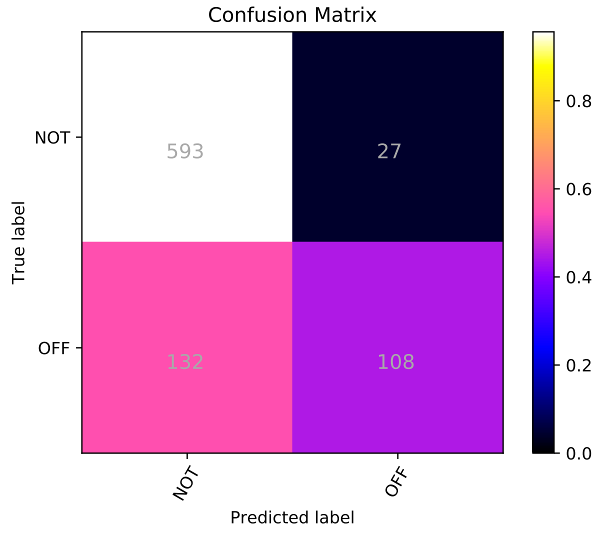


Figure 2: Sub-task A, Ensemble of LR-NB-RF

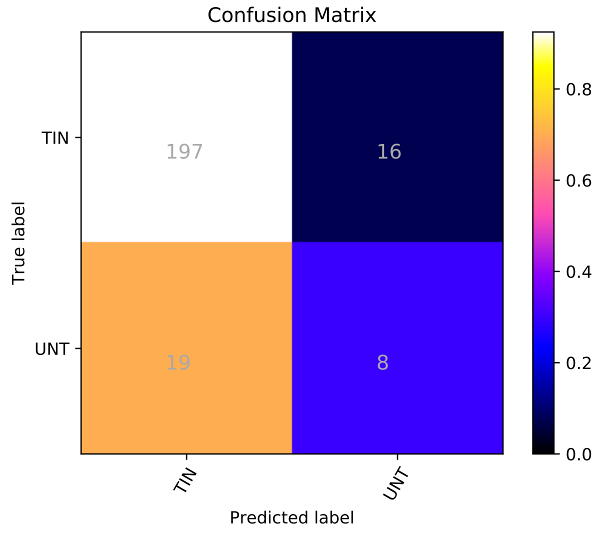


Figure 3: Sub-task B, Naive Bayes - TFIDF - Lemmatization

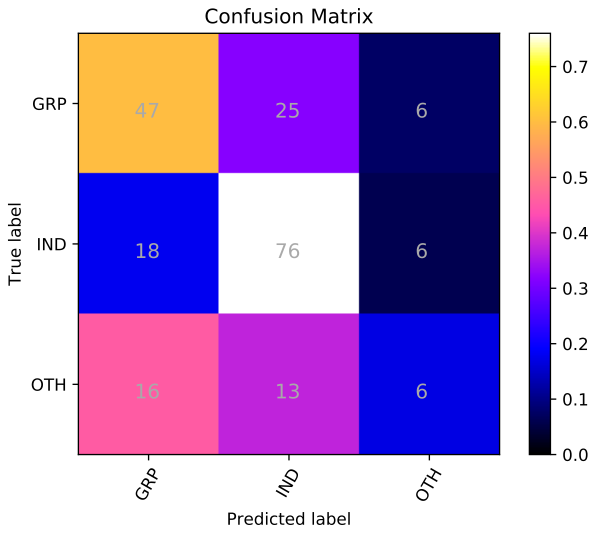


Figure 4: Sub-task C, Logistic Regression - Count - Stopwords Removal

5 Conclusion

This paper describes our offensive tweets identification and categorization system that was built in the framework of SemEval-2019 Task 6. We used a brute-force search technique to find the best model that could be generated from a list of prepocessing techniques, feature extraction models and classifiers and got an F1-sore of 0.728 in subtask A, 0.6161 in subtask B and 0.5093 in subtask C. In future work, we aim to focus more on word embedding features by concatenating all 3 word vector models and experiment with character-level/sentence-level models.

References

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 467–472.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*.

Martin F Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Alexandra Schofield and Thomas Davidson. 2017. Identifying Hate Speech in Social Media. *XRDS: Crossroads, The ACM Magazine for Students*, 24(2):56–59.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.

SemEval-2019 Task 4: Hyperpartisan News Detection

Johannes Kiesel[1] **Maria Mestre**[2,3] **Rishabh Shukla**[2] **Emmanuel Vincent**[2]
Payam Adineh[1] **David Corney**[4] **Benno Stein**[1] **Martin Potthast**[5]

[1] Bauhaus-Universität Weimar
<first>.<last>@uni-weimar.de

[2] Factmata Ltd.
<first>.<last>@factmata.com

[3] mariarmestre@gmail.com
[4] dpacorney@gmail.com

[5] Leipzig University
martin.potthast@uni-leipzig.de

Abstract

Hyperpartisan news is news that takes an extreme left-wing or right-wing standpoint. If one is able to reliably compute this meta information, news articles may be automatically tagged, this way encouraging or discouraging readers to consume the text. It is an open question how successfully hyperpartisan news detection can be automated, and the goal of this SemEval task was to shed light on the state of the art. We developed new resources for this purpose, including a manually labeled dataset with 1,273 articles, and a second dataset with 754,000 articles, labeled via distant supervision. The interest of the research community in our task exceeded all our expectations: The datasets were downloaded about 1,000 times, 322 teams registered, of which 184 configured a virtual machine on our shared task cloud service TIRA, of which in turn 42 teams submitted a valid run. The best team achieved an accuracy of 0.822 on a balanced sample (yes : no hyperpartisan) drawn from the manually tagged corpus; an ensemble of the submitted systems increased the accuracy by 0.048.

1 Introduction

Yellow journalism has established itself in social media, nowadays often linked to phenomena like clickbait, fake news, and hyperpartisan news. Clickbait has been its first "success story" (Potthast et al., 2016): When the viral spreading of pieces of information was first observed in social networks, some investigated how to manufacture such events for profit. Unlike for "natural" viral content, however, readers had to be directed to a web page containing the to-be-spread information alongside paid-for advertising, so that only teasers and not the information itself could be shared. Then, to maximize their virality, data-driven optimization revealed that teaser messages which induce curiosity, or any other kind of strong emotion, spread best. The many forms of such teasers that have emerged since are collectively called clickbait. New publishing houses arose around viral content, which brought clickbait into the mainstream. Traditional news publishers, struggling for their share of the attention market that is a social network, adopted clickbait into their toolbox, too, despite its violation of journalistic codes of ethics.

The content spread using clickbait used to be mostly harmless trivia—entertainment and distraction to some, spam to others—, but in the wake of the 2016 United States presidential election, "fake news" came to widespread public attention. While certainly not a new phenomenon in yellow journalism, its viral success on social media was a surprise to many. Part of this success was then attributed to so-called hyperpartisan news publishers (Bhatt et al., 2018), which report strongly in favor of one political position and in fierce disagreement with its opponents. Clinging to hyperpartisanship often entails stretching the truth, if not breaking it with fake news, whose highly emotional content makes them spread exceptionally fast, like clickbait.

Given the hype surrounding fake news, activists, industry, and research are now paying a lot of attention to mitigating the problem, such as trying to check facts in news items. Clickbait and hyperpartisan news, however, have been less studied. In previous work, we sought to help close this gap from both ends: for clickbait detection (Potthast et al., 2016), part of our group created a large-scale evaluation dataset (Potthast et al., 2018b) and set up an ongoing competition for the best detection approach (Potthast et al., 2018a). For hyperpartisan news detection (Potthast et al., 2018c), we teamed up to follow a similar approach that led to the Hyperpartisan News Detection task at SemEval-2019. This paper reports on the results of this task.

2 Task Definition

We define hyperpartisan news detection as follows:

> Given the text and markup of an online news article, decide whether the article is hyperpartisan or not.

Hyperpartisan articles mimic the form of regular news articles, but are one-sided in the sense that opposing views are either ignored or fiercely attacked. We deliberately disregard the distinction between left and right, since previous work has found that, in hyperpartisan form, both are more similar to each other in terms of style than either are to the mainstream (Potthast et al., 2018c). The challenge of this task is to unveil the mimicking and to detect the hyperpartisan language, which may be distinguishable from regular news at the levels of style, syntax, semantics, and pragmatics.

3 Data

Our focus is on news articles published online, and we provide two datasets with this task. One has 1,273 articles, each labeled manually, while the second, larger dataset of 754,000 articles is labeled in a semi-automated manner via distant supervision at the publisher level. These datasets are further split into public and private sets. We released the public set for the model training, tuning, and evaluation,[1] while the unreleased private set is used to enable blind, cloud-based evaluation.

As online news articles are published mainly in the HTML format, both datasets use a unified HTML-like format (see Figure 1). We restricted the markup for the article content to paragraphs (`<p>`), links (`<a>`), and quotes (`<q>`). We distinguished *internal* links to the other pages of the same domain, from which we removed the href-attribute value to avoid classifiers fitting to them; and links to *external* domains, for which we kept the attribute. An XML schema that exactly specifies the format is distributed along the datasets.

3.1 Dataset Annotated By Article

We gathered a crowdsourced dataset of 1,273 articles, each labeled manually by 3 annotators (Vincent and Mestre, 2018). These articles were published by active hyperpartisan and mainstream websites and were all assured to contain political news. Annotators were asked to rate each article's bias on the following 5-point Likert scale:

[1]https://doi.org/10.5281/zenodo.1489920

1. No hyperpartisan content
2. Mostly unbiased, non-hyperpartisan content
3. Not Sure
4. Fair amount of hyperpartisan content
5. Extreme hyperpartisan content

We removed all articles from the dataset with low agreement score and the aggregated rating of "not sure" (see Vincent and Mestre for more details). We then binarized the labels to hyperpartisan (average rating of 4 or 5) and not (average rating of 1 or 2). The final by-article set achieved an inter-annotator agreement of 0.5 Krippendorff's alpha. Of the remaining 1,273 articles, 645 were published as a training dataset, whereas the other 628 (50% hyperpartisan and 50% not) were kept private for the evaluation. To ensure that classifiers could not profit from overfitting to publisher style, we made sure there was no overlap between the publishers of the articles between these two sets.

3.2 Dataset Annotated By Publisher

To allow for methods that require huge amounts of training data, we compiled a dataset of 754,000 articles, each labeled as per the bias of their respective publisher. To create this dataset, we cross-checked two publicly available news publisher bias lists compiled by media professionals from BuzzFeed news[2] and Media Bias Fact Check.[3] The former was created by BuzzFeed journalists as a basis for a news article, whereas the latter is Media Bias Fact Check's main product. While both lists contain several hundred news publishers, they disagree only for nine, which we removed from our dataset.

We then crawled, archived, and post-processed the articles available on the publishers' web sites and Facebook feeds. We archived all articles using a specialized tool (Kiesel et al., 2018) that removes pop-overs and similar things preventing the article content from being loaded. After filtering out publishers that did not mainly publish political articles or had no political section to which we could restrict our crawl, we were left with 383 publishers. For each of the publishers' web sites we wrote a content-wrapper to extract the article content and relevant meta data from the HTML DOM. We then removed all articles that were too short to contain news,[4] that are not written in English,

[2]https://github.com/BuzzFeedNews/2017-08-partisan-sites-and-facebook-pages

[3]https://mediabiasfactcheck.com

[4]Based on manual inspection of a hundred short articles, we set the threshold to 40 words.

```
<article id="0182515" published-at="2007-01-22" title="They’re crumbling">
<p>What a pleasant surprise to see Jacques Leslie, a journalist and real expert on
dams, with a long <a href="http://www.nytimes.com/2007/01/22/opinion/22leslie.2.html
?ex=1327122000&amp;amp;en=42caf99f05e4cba8&amp;amp;ei=5090&amp;amp;partner=
rssuserland&amp;amp;emc=rss" type="external">op-ed</a> on the hallowed pages of the
New York Times.  Leslie, author of <a href="" type="internal">Deep Water:  The Epic
Struggle Over Dams, Displaced People and the Environment</a>, highlights the threat
posed by poorly maintained and increasingly failing dams around the country:</p>
<p>Unlike, say, waterways and sanitation plants, a majority of dams - 56 percent of
those inventoried - are privately owned, which is one reason dams are among the
country’s most dangerous structures.  Many private owners can’t afford to repair
aging dams; some owners go so far as to resist paying by tying up official repair
demands in court or campaigning to weaken state dam safety laws.</p>
<p>Kinda makes you want to find out what is upstream.</p> </article>
```

Figure 1: Example of a non-hyperpartisan article in our dataset. An archived version of the original article is available at https://web.archive.org/web/20121006194050/https://grist.org/article/remember-the-dams/.

or that contain obvious encoding errors. The final dataset consisted of 754,000 articles, split into a public training set (600,000 articles), a public validation set (150,000 articles) and a non-public test set (4,000 articles). Like for the by-article dataset, we ensured that there is no overlap of publishers between the sets. Each set consists of 50% articles from non-hyperpartisan publishers and 50% articles from hyperpartisan publishers, the latter again being 50% from left-wing and 50% from right-wing publishers.

4 Fairness and Reproducibility

In this shared task, we asked participants to submit their software instead of just its run output. The submissions were executed at our site on the test data, enabling us to keep the test data entirely secret. This has two important advantages over traditional shared task setups: first, software submission gives rise to blind evaluation; and second, it maximizes the replicability and the reproducibility of each participant's approach. To facilitate software submission and to render it feasible in terms of work overhead and flexibility for both participants and organizers, we employ the TIRA Integrated Research Architecture (Potthast et al., 2019).

A shortcoming of traditional shared task setups is that typically the test data are shared with participants, albeit without ground truth. Although participants in shared tasks generally exercise integrity and do not analyze the test data other than running their software on it, we have experienced cases to the contrary. Such problems particularly arise in shared tasks where the stakes are higher than usual; when monetary incentives are offered or winning results in high visibility. A partial workaround is to share the test data only very close to the final submission deadline, minimizing analysis opportunities. But if sharing the test data is impossible for reasons of sensibility and proprietariness, or because the ground truth can be easily reverse-engineered, a traditional shared task cannot be held.

Another shortcoming of traditional shared tasks (and many computer science publications in general) is their lack of reproducibility. Although sharing the software underlying experiments as well as the trained models is easy, and although it would greatly aid reproducibility, this is still rare. Typically, all that remains after a shared task are the papers and datasets published. Given that shared tasks often establish a benchmark for the task in question, acting normative for future evaluations, this outcome is far from optimal and comparably wasteful. All of the above can be significantly improved upon by asking participants not to submit their software's run output, but the software itself. However, this entails a significant work overhead for organizers, especially for larger tasks.

In order to mitigate the work overhead, we employ TIRA. In a nutshell, TIRA implements evaluation as a service in the form of a cloud-based evaluation platform. Participants deploy their software into virtual machines hosted at TIRA's cloud, and then remotely control the machines and the software within, executing it on the test data. The test data are available only within the cloud, and made accessible on demand so that participants cannot access it directly. At execution time, the virtual machine is disconnected from the internet, copied, and only the copy gets access to the test data. Once the automatically executed software terminates, its run output is saved and the virtual machine copy is destroyed so as to prevent data leaks. This way, all submitted pieces of software can be archived in working condition, and be re-evaluated at a later time, even on new datasets.

5 Participating Systems

This task attracted a very diverse and interesting set of solutions from the participating teams. The teams employed very different sets of features, a wide variety of classifiers, and also employed the large by-publisher dataset in different ways. Around half of the submissions used hand-crafted features. In the following, we give an overview of the submitted approaches. For a more readable and condensed form, we only use the team names here, which were chosen from fictional journalistic characters or entities (see Table 1 for references).

5.1 Features

The teams that participated in this task employed a variety of features, including standard word n-grams (also unigrams, i.e., bag-of-words), word embeddings, stylometric features, HTML features like the target of hyperlinks, and a meta data feature in the form of the publication date.

N-Grams Most teams that used hand-crafted features also included word n-grams: Pioquinto Manterola and Tintin used them as their only features. Character and part-of-speech n-grams were, for example, used by Paparazzo.

Word embeddings Many teams integrated word embeddings into their approach. Frequently used were Word2Vec, fastText, and GloVe. Noticeably, Tom Jumbo Grumbo relied exclusively on them. Bertha von Suttner relied on ELMo embeddings (Peters et al., 2018), which have the advantage of modeling polysemy. Where the aforementioned word embeddings all rely on neural networks, Doris Martin employed a document representation based on word clusters as part of their approach.

BERT (Devlin et al., 2018), which jointly conditions on both left and right context in all layers, is a rather new technique that was used by several teams. Peter Parker directly applied a freely available pre-trained BERT model to the task, whereas Howard Beale and Clint Buchanan trained their own BERT models on the by-publisher dataset and then performed fine-tuning on the by-article dataset. Despite the fine-tuning, Howard Beale reported overfitting issues for this strategy. Going one step further, Jack Ryder and Yeon Zi integrated BERT in their neural network architectures.

Stylometry Many teams used stylometric features including punctuation and article structure (Steve Martin, Spider Jerusalem, Fernando Pessa, Ned Leeds, Carl Kolchak, Orwellian Times), readability scores (Ned Leeds, Pistachon, Steve Martin, Orwellian Times, D X Beaumont), or psycholinguistic lexicons (Ned Leeds, Spider Jerusalem, Steve Martin, Pistachon). Borat Sagdiyev employed a self-compiled list of trigger words that contains mostly profanities. They noticed that such words are used more often in hyperpartisan articles.

Emotionality Several teams used sentiment and emotion features, either based on libraries (Borat Sagdiyev, Steve Martin, Carl Kolchak) or lexicons (Spider Jerusalem, D X Beaumont). Notably, Kermit the Frog uses sentiment detection only. Vernon Fenwick and D X Beaumont used subjectivity and polarity metrics as features.

Named entities Borat Sagdiyev used named entity types as features. In preliminary tests only the type of "nationalities or religious and political groups" was found to be predictive.

Quotations A few teams treated quotations separately. Whereas Spider Jerusalem and Borat Sagdiyev created separate features from quotations, the Ankh Morpork Times filtered them out for not necessarily representing the views of the author.

Hyperlinks Only few teams considered hyperlinks. Both Borat Sagdiyev and Steve Martin used external lists of partisan web pages to count how often an article links to partisan and non-partisan pages. They assume that articles tend to link other articles on the same side of the political spectrum.

Publication date Based on the conjecture that months around American elections could see more hyperpartisan activity, Borat Sagdiyev used the publication month and year as separate features.

5.2 Classifiers

While many different classifiers were used overall, neural networks were the most frequent, which mirrors the current trend in text classification.

The most popular type of neural networks among the participants were convolutional ones (CNNs), which employ convolving filters over neighboring words. Many teams cited the architecture by Kim (2014). Xenophilius Lovegood added a second layer to their CNN in order to encode more information about the articles, using both available and custom-learned embeddings. While Pioquinto Manterola experimented with a CNN, it suffered from overfitting and was thus not used for the final submission. Peter Brinkmann built a submission using available embeddings. Brenda Starr combined a CNN with a sentence-level bidirectional recurrent neural network and an attention mechanism to a complex architecture. A similar approach was employed by the Ankh Morpork Times. An ensemble of three CNN-based models was used by Bertha von Suttner. Steve Martin used a character bigram CNN as part of their approach.

Next to CNNs, long short term memory networks (LSTM) were employed by Kit Kittredge and Miles Clarkson. The latter extended the network with an attention model. Moreover, Joseph Rouletabille used the hierarchical attention network of Yang et al. (2016).

Besides neural networks, a wide variety of classifiers were used. A few teams opted for SVMs (e.g., the Orwellian Times), others for random forests (e.g., Fernando Pessa), linear models (e.g., Pistachon), the Naive Bayes model (e.g., Carl Kolchak), XGBOOST (Clark Kent), Maxent (Doris Martin), and rule-based models (Harry Friberg). Morbo used ULMFit (Howard and Ruder, 2018) to adapt a language model pre-trained on Wikipedia articles to the articles and classes of this task.

5.3 Usage of the By-publisher Dataset

The submitted systems can also be distinguished by whether and how they used the large, distantly-supervised by-publisher dataset. Though much larger than the by-article set, its labels are noisy, whereas the opposite holds for the by-article dataset. One of the key challenges faced by many teams was how to train a powerful expressive model on the smaller dataset without overfitting. Most teams made use of the larger dataset in some form or another. A challenge faced by some of the teams was that the test split of the by-article dataset was balanced between classes, whereas the corresponding training dataset was not.

Several systems trained the whole or part of their system on the by-publisher dataset. Some extracted features like n-grams (e.g., Sally Smedley), word clusters (Doris Martin), or neural network word embeddings (e.g., Clint Buchanan). Others used the larger dataset to perform hyperparameter search (e.g., Miles Clarkson). Many teams trained their models using the by-publisher dataset only (Pistachon, Joseph Rouletabille, Xenophilius Lovegood, Peter Brinkmann, and Kit Kittredge).

To reduce the noise in the distantly-supervised data, some teams used only a subset of it. Yeon Zi, Borat Sagdiyev and the Anhk Morpork Times fitted a model on the by-article dataset and ran it on the by-publisher one: the articles of the by-publisher dataset that were misclassified by this model, were presumed to be noisy and filtered out.

6 Results

A total of 42 teams completed the task, representing more than twenty countries between them, including India, China, the USA, Japan, Vietnam, and many European countries. Table 1 shows the accuracy, precision, recall, and F_1 score for each team, sorted by accuracy. This task used accuracy as the main metric to represent a filtering scenario. The accuracy scores ranged from 0.462 up to 0.822.

The results show a range of trade-offs between precision and recall and the resulting F_1 scores. The highest F_1 was 0.821 with a precision of 0.815 and a recall of 0.828; the highest precision was 0.883 with a recall of 0.672 (F_1: 0.763); and the highest recall was 0.971 with a relatively low precision of 0.542 (F_1: 0.696).

6.1 Methods Used by the Top Teams

While the winning team, Bertha von Suttner, used deep learning (sentence-level embeddings and a convolutional neural network) the second-placed team, Vernon Fenwick, took a different approach and combined sentence embeddings with more domain-specific features and a linear model. Out of the top five teams, only two used "pure" deep learning models of neural networks without any domain-specific, hand-crafted features, showing no single method has a clear advantage over others.

Bertha von Suttner used a model based on ELMo embeddings (Peters et al., 2018) and trained on the by-article dataset. After minimal preprocessing, a pre-trained ELMo was applied onto each token of each sentence, and then averaged, to obtain average sentence embeddings. The sentence embeddings were later passed through a CNN, batch-normalized, followed by a dense layer and a sigmoid function to obtain the final probabilities. The final model was an ensemble of the 3 best-performing models of a 10-fold cross-validation. The authors tried to include the by-publisher dataset, but found in their preliminary tests no approach to profit from the large data.

Submission			By-article dataset					By-publisher dataset				
Team name	Authors	Code	Rank	Acc.	Prec.	Recall	F_1	Rank	Acc.	Prec.	Recall	F_1
Bertha von Suttner	Jiang et al.	🔗	1	**0.822**	0.871	0.755	0.809	8	0.643	0.616	0.762	0.681
Vernon Fenwick	Srivastava et al.		2	0.820	0.815	0.828	**0.821**					
Sally Smedley	Hanawa et al.		3	0.809	0.823	0.787	0.805	11	0.625	0.640	0.571	0.603
Tom Jumbo Grumbo	Yeh et al.	🔗	4	0.806	0.858	0.732	0.790	13	0.619	0.592	0.762	0.667
Dick Preston	Isbister and Johansson		5	0.803	0.793	0.818	0.806	27	0.514	0.520	0.352	0.420
Borat Sagdiyev	Palić et al.		6	0.791	**0.883**	0.672	0.763	19	0.592	0.644	0.412	0.502
Morbo	Isbister and Johansson		7	0.790	0.772	0.822	0.796	16	0.601	0.587	0.679	0.630
Howard Beale	Mutlu et al.		8	0.783	0.837	0.704	0.765	9	0.641	0.606	0.806	0.692
Ned Leeds	Stevanoski and Gievska		9	0.775	0.865	0.653	0.744	22	0.573	0.546	0.857	0.667
Clint Buchanan	Drissi et al.	🔗	10	0.771	0.832	0.678	0.747					
Yeon Zi	Lee et al.		11	0.758	0.744	0.787	0.765	5	0.663	0.635	0.766	0.694
Tony Vincenzo	Staykovski		12	0.750	0.764	0.723	0.743					
Paparazzo	Nguyen et al.	🔗	13	0.747	0.754	0.732	0.743	24	0.530	0.530	0.541	0.535
Steve Martin	Joo and Hwang		14	0.745	0.853	0.592	0.699	18	0.597	0.625	0.483	0.545
Eddie Brock	Šajatović et al.		15	0.744	0.782	0.675	0.725	10	0.631	0.681	0.491	0.571
Ankh Morpork Times	Almendros et al.		16	0.742	0.811	0.631	0.710	21	0.588	0.646	0.389	0.486
Spider Jerusalem	Alabdulkarim and Alhindi	🔗	17	0.742	0.814	0.627	0.709					
Carl Kolchak	Chen et al.		18	0.739	0.729	0.761	0.745					
Doris Martin	Agerri	🔗	19	0.737	0.754	0.704	0.728					
Pistachon	Saleh et al.		20	0.729	0.724	0.742	0.733	15	0.608	0.638	0.499	0.560
Joseph Rouletabille	Moreno et al.		21	0.725	0.788	0.615	0.691	2	0.680	0.640	0.827	**0.721**
Fernando Pessa	Cruz et al.	🔗	22	0.717	0.806	0.570	0.668	17	0.600	0.585	0.681	0.630
Pioquinto Manterola	Sengupta and Pedersen	🔗	23	0.704	0.741	0.627	0.679					
Miles Clarkson	Zhang et al.		24	0.683	0.745	0.557	0.638	6	0.652	0.612	0.832	0.705
Xenophilius Lovegood	Zehe et al.		25	0.675	0.619	0.914	0.738	4	0.663	0.632	0.781	0.699
Orwellian Times	Knauth		26	0.672	0.654	0.729	0.690	23	0.537	0.530	0.658	0.587
Tintin	Bestgen		27	0.656	0.642	0.707	0.673	1	**0.706**	0.742	0.632	0.683
D X Beaumont	Amason et al.		28	0.653	0.597	0.939	0.730					
Jack Ryder	Shaprin et al.		29	0.646	0.646	0.646	0.646	7	0.645	0.600	**0.869**	0.710
Kermit the Frog	Anthonio and Kloppenburg		30	0.621	0.582	0.860	0.694	20	0.589	0.575	0.681	0.623
Billy Batson	Kreutz et al.		31	0.615	0.568	0.962	0.714					
Peter Brinkmann	Färber et al.	🔗	32	0.602	0.560	0.955	0.706	28	0.497	0.496	0.344	0.406
Anson Bryson	Stiff and Medero		33	0.592	0.720	0.303	0.426					
Sarah Jane Smith	Chakravartula et al.		34	0.591	0.554	0.933	0.695	14	0.612	0.586	0.765	0.664
Kit Kittredge	Cramerus and Scheffler		35	0.578	0.547	0.908	0.683					
Brenda Starr	Papadopoulou et al.		36	0.575	0.542	**0.971**	0.696	3	0.664	0.627	0.807	0.706
Harry Friberg	Afsarmanesh et al.		37	0.565	0.537	0.949	0.686					
Robin Scherbatsky	Marx and Akut		38	0.551	0.542	0.662	0.596	25	0.524	**0.822**	0.062	0.116
Clark Kent	Gupta et al.	🔗	39	0.548	0.683	0.178	0.283	26	0.519	0.565	0.170	0.261
Murphy Brown	Sen and Jiang		40	0.529	0.518	0.822	0.635	12	0.623	0.615	0.659	0.636
Peter Parker	Ning et al.		41	0.503	0.502	0.771	0.608					
John King	Bansal et al.		42	0.462	0.460	0.443	0.451					

Table 1: For each team and dataset, the performance of the submission that reached the highest accuracy is shown. If a team published their code, the 🔗 links to the respective repository. We forked all repositories for archival.[6]

The second and third best teams used linear models as their main predictor and embeddings as features, training on the by-article dataset only. Vernon Fenwick extracted sentence embeddings with the Universal Sentence Encoder (USE) (Cer et al., 2018), while Sally Smedley used BERT to generate contextual embeddings. Both teams also employed hand-crafted, domain-specific features. Vernon Fenwick extracted article-level and sentence-level polarity, bias, and subjectivity, among others, while Sally Smedley used the by-publisher dataset to extract key discriminative phrases, which they later looked up in the training data.

6.2 Overall Insights

The results reveal several insights into the suitability of different features and approaches for the task of hyperpartisan news detection.

Word-embeddings have been reported to be a very efficient feature by many teams. Tom Jumbo Grumbo achieved an accuracy of 0.806 with GloVe embeddings and a classifier trained on the by-article dataset. The application of a pre-trained BERT model by Peter Parker performed very poorly (acc. 0.503). However, the same BERT embeddings were used for great effect by Sally Smedley, using techniques like word-dropout and informative phrase identification (acc. 0.809).

[6]https://github.com/hyperpartisan-news-challenge

Also standard word n-grams were found to be suitable for the task, though not as strong as embeddings. While n-grams where used in several well-performing approaches, Pioquinto Manterola reached an accuracy of 0.704 with unigrams alone.

Several teams reported an increase in accuracy through sentiment or similar features (e.g., Borat Sagdiyev). Kermit the Frog used sentiment detection alone to reach an accuracy of 0.621.

Besides textual features, a few teams also analyzed HTML and article meta-features. Borat Sagdiyev performed a detailed analysis in this regard, which helped them to achieve the highest precision of all teams. For example, they found that both the publication date and the number of links to known hyperpartisan pages could each improve the overall accuracy by about 0.01 to 0.02.

Of the top teams, only Sally Smedley used the by-publisher dataset, and only to select n-grams. Based on the reports of several teams, the utilization of this dataset thus seems more difficult than we expected. We conjecture that this is due to the mis-classification of what should be the most informative articles: non-hyperpartisan articles from mainly hyperpartisan publishers, and hyperpartisan articles from non-hyperpartisan publishers. These articles are especially suited to distinguish features that identify hyperpartisanship from features that identify publisher style. While we assumed that the advantages of big data would outweigh this drawback, the results suggest that it might be more worthwhile to put effort in larger datasets where each article is annotated separately. Still, some teams managed to use the by-publisher dataset as a large dataset of in-domain texts. For example, Clint Buchanan reported that pre-training embeddings on the by-publisher dataset increased the accuracy of their system on the by-article dataset.

Moreover, the ranking of teams for the two test datasets is quite different. Bertha von Suttner, who ranked first for by-article, reached only rank eight for the by-publisher dataset. Conversely, Tintin, who optimized for by-publisher, ranked first there but only 27th for the by-article dataset. This discrepancy highlights the unexpected large differences between the datasets.

7 Meta-Classification Task

Inspired by successes of meta classifiers in past SemEval tasks (e.g., Hagen et al. (2015)), we enabled and encouraged participants to devise meta

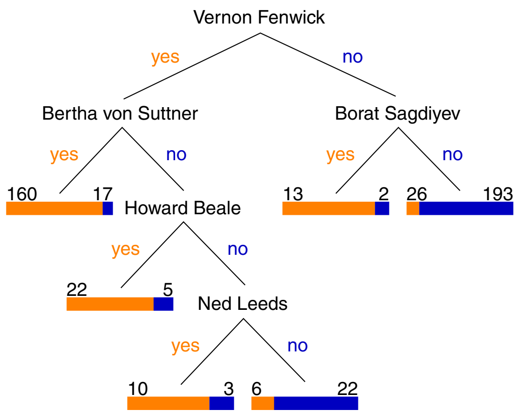


Figure 2: Meta-classification decision tree J48-M10 learned on the predictions of the submitted systems (hyperpartisan: yes or no; by-article dataset). The numbers show the training class-distribution at the leafs.

classifiers that learn from the classifications of the submitted approaches. For this meta-classification task, we split the test datasets further into new training (66%) and test sets (33%). We again made sure that there are an equal amount of non-hyperpartisan and hyperpartisan articles, as well as an equal share of left-wing and right-wing articles within the hyperpartisan sets. Furthermore, we again assured that no publisher had articles in both the training and the test sets. An instance in these datasets corresponds to the classifications (hyperpartisan or not) of the best-performing software of each team (42 classifications for the by-article dataset and 30 for the by-publisher one) of one article from the original test data.

We provide two simple classification systems for baselines, majority voting and an out-of-the-box decision tree, which both outperform the best single submitted software and which were both outperformed by the meta-classifiers submitted. Majority voting refers to a system that outputs the classification (hyperpartisan or not) that the most base classifiers selected. As it does not learn a decision boundary, it is—strictly speaking—not a meta classifier. For the decision tree, we used the J48 implementation of WEKA (Frank et al., 2016). We tested two variants: standard settings (*J48-M2*) and restricting leaf nodes to contain at least 10 articles (*J48-M10*) to force a simpler decision tree. Simpler trees often generalize better to unseen data.

Figure 2 shows the J48-M10 tree for the by-article dataset. For every leaf of the tree, more than 75% of the corresponding training articles are from the same class. This shows that even with as few as 5 decision nodes, the training set

Team or system name	Acc.	Prec.	Recall	F_1
Fernando Pessa	**0.899**	0.895	**0.904**	**0.900**
Spider Jerusalem	**0.899**	0.903	0.894	0.899
Majority Vote	0.885	0.892	0.875	0.883
J48-M10	0.880	**0.916**	0.837	0.874
J48-M2	0.856	0.863	0.846	0.854
Bertha von Suttner alone	0.851	0.901	0.788	0.841

Table 2: Accuracy, precision, recall, and F_1-measure for the by-article meta learning test dataset.

could be fitted reasonably well. The meta classifier was thus able to use the submitted systems as predictive and distinct features, which shows that some submitted systems performed well on some articles where other systems did not and vice versa. Even more, the 5 systems employed by the meta-classifier are all within the top 10 systems of the task, which shows that there is considerable variation even among the top performers. This is reasonable, given the variety of approaches used.

In addition to our approaches, two teams submitted their own classifiers in the short time span they had. Fernando Pessa used a random forest classifier trained on the single predictions as well as the average vote. Spider Jerusalem used a weighted majority voting algorithm, where they weighted each single prediction by the precision of the respective classifier on the training set.

Table 2 shows the performance of the approaches on the meta learning test dataset. Note that the best single system, Bertha von Suttner, reaches an increased accuracy of 0.851 on the meta learning test set. This is due to variations in the small dataset. Still, all ensemble approaches reach a higher accuracy. The majority voting approach reaches an accuracy of 0.885, and thus outperforms the J48 classifiers. This is somewhat surprising, but shows that there is a lot to gain by integrating also the systems that performed less well—team Fernando Pessa came to a similar insight in their paper (Cruz et al., 2019). The approaches of the two participants performed very similar, despite their methodological differences, and outperformed the majority vote. They managed to achieve an accuracy 0.048 points above Bertha von Suttner and therefore a considerable increase in performance.

We also repeated the experiments for the by-publisher dataset, but could not produce decisive results there, yet. We assume that this is due to most teams focusing on the other dataset and both datasets being more different than expected.

8 Conclusion

This paper reports on the setup, participation, results, and insights gained from the first task in hyperpartisan news detection, hosted as Task 4 at SemEval-2019. We detailed the construction of both a manually annotated dataset of 1,273 articles as well as a large dataset of 754,000 articles, compiled using distant supervision. Moreover, it provides a systematic overview of the 34 papers submitted by the participants, insights gathered from single teams, by comparing their approaches, and by an ad-hoc meta classification.

Through the use of TIRA (Potthast et al., 2019), we were able to establish a blind evaluation setup, so that future approaches can be compared on same grounds. For this, we continue to accept new approaches in ongoing submissions.[7] Moreover, through the use of TIRA we can directly evaluate the submitted approaches on new datasets for hyperpartisan news detection, provided they are formatted like the datasets presented here.

Very promising results were achieved during the task, with accuracy values above 80% on a balanced test set—and even up to 90% using meta classification on all submissions. Like in many other NLP tasks, word embeddings could be used to great effect, but hand-crafted features also performed well. The differences between the two employed datasets were larger than anticipated, which suggests a focus on by-article annotations in the future. A larger dataset of this kind will probably assist in improving the accuracy of future models even beyond the already very good level.

It thus seems that hyperpartisan news detection is already sufficiently developed to take the next step and demand human-understandable explanations from the approaches. The most obvious use cases of hyperpartisan news detectors are for filtering articles, which always requires a careful handling to avoid unwarranted censorship. Especially in the current political climate, it therefore seems necessary that hyperpartisanship detectors not only reach a high accuracy, but also reveal their reasoning.

Acknowledgements

Our thanks go out to all participating teams; your contributions made this task a success. We hope we have been able to do your work justice, and are looking forward to doing so in the future. Our special thanks go out to the SemEval organizers for providing perfect organizational support.

[7]https://webis.de/events/semeval-19/

References

Nazanin Afsarmanesh, Jussi Karlgren, Peter Sumbler, and Nina Viereckel. 2019. Team Harry Friberg at SemEval-2019 Task 4: Identifying Hyperpartisan News through Editorially Defined Metatopics. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Rodrigo Agerri. 2019. Doris Martin at SemEval-2019 Task 4: Hyperpartisan News Detection with Generic Semi-supervised Featuresl. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Amal Alabdulkarim and Tariq Alhindi. 2019. Spider-Jerusalem at SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Carla Perez Almendros, Luis Espinosa Anke, and Steven Schockaert. 2019. Cardiff University at SemEval-2019 Task 4: Linguistic Features for Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Evan Amason, Jake Palanker, Mary Clare Shen, and Julie Medero. 2019. Harvey Mudd College at SemEval-2019 Task 4: The D.X. Beaumont Hyperpartisan News Detector. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Talita Anthonio and Lennart Kloppenburg. 2019. Team Kermit-the-frog at SemEval-2019 Task 4: Bias Detection Through Sentiment Analysis and Simple Linguistic Features. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Yves Bestgen. 2019. Tintin at SemEval-2019 Task 4: Detecting Hyperpartisan News Article with only Simple Tokens. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Shweta Bhatt, Sagar Joglekar, Shehar Bano, and Nishanth Sastry. 2018. Illuminating the ecosystem of partisan websites. In *Proceedings of the 27th International Conference on World Wide Web Companion*, WWW '18 Companion. International World Wide Web Conferences Steering Committee.

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder.

Nikhil Chakravartula, Vijayasaradhi Indurthi, and Bakhtiyar Syed. 2019. Fermi at SemEval-2019 Task 4: The sarah-jane-smith Hyperpartisan News Detector. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Celena Chen, Celine Park, Jason Dwyer, and Julie Medero. 2019. Harvey Mudd College at SemEval-2019 Task 4: The Carl Kolchak Hyperpartisan News Detector. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Rebekah Cramerus and Tatjana Scheffler. 2019. Team Kit Kittredge at SemEval-2019 Task 4. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

André Cruz, Gil Rocha, Rui Sousa-Silva, and Henrique Lopes Cardoso. 2019. Team Fernando-Pessa at SemEval-2019 Task 4: Back to Basics in Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, abs/1810.04805.

Mehdi Drissi, Pedro Sandoval Segura, Vivaswat Ojha, and Julie Medero. 2019. Harvey Mudd College at SemEval-2019 Task 4: The Clint Buchanan Hyperpartisan News Detector. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Michael Färber, Agon Qurdina, and Lule Ahmedi. 2019. Team Peter Brinkmann at SemEval-2019 Task 4: Detecting Biased News Articles Using Convolutional Neural Networks. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Eibe Frank, Mark A. Hall, and Ian H. Witten. 2016. *Data Mining: Practical Machine Learning Tools and Techniques*, 4th edition, chapter The WEKA Workbench. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Viresh Gupta, Baani Leen Kaur Jolly, Ramneek Kaur, and Tanmoy Chakraborty. 2019. Clark Kent at SemEval-2019 Task 4: Stylometric Insights into Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Matthias Hagen, Martin Potthast, Michel Büchner, and Benno Stein. 2015. Webis: An Ensemble for Twitter Sentiment Detection. In *9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 582–589. Association for Computational Linguistics.

Kazuaki Hanawa, Shota Sasaki, Hiroki Ouchi, Jun Suzuki, and Kentaro Inui. 2019. The Sally Smedley Hyperpartisan News Detector at SemEval-2019 Task 4. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. *CoRR*, abs/1801.06146.

Tim Isbister and Fredrik Johansson. 2019. Dick-Preston and Morbo at SemEval-2019 Task 4: Transfer Learning for Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Ye Jiang, Johann Petrak, Xingyi Song, Kalina Bontcheva, and Diana Maynard. 2019. Team Bertha von Suttner at SemEval-2019 Task 4: Hyperpartisan News Detection using ELMo Sentence Representation Convolutional Network. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Youngjun Joo and Inchon Hwang. 2019. Steve Martin at SemEval-2019 Task 4: Ensemble Learning Model for Detecting Hyperpartisan News. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Johannes Kiesel, Florian Kneist, Milad Alshomary, Benno Stein, Matthias Hagen, and Martin Potthast. 2018. Reproducible Web Corpora: Interactive Archiving with Automatic Quality Assessment. *Journal of Data and Information Quality (JDIQ)*, 10(4):17:1–17:25.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jürgen Knauth. 2019. Orwellian-times at SemEval-2019 Task 4: A Stylistic and Content-based Classifier. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Nayeon Lee, Zihan Liu, and Pascale Fung. 2019. Team yeon-zi at SemEval-2019 Task 4: Hyperpartisan News Detection by De-noising Weakly-labeled Data. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Jose G. Moreno, Yoann Pitarch, Karen Pinel-Sauvagnat, and Gilles Hubert. 2019. Rouletabille at SemEval-2019 Task 4: Neural Network Baseline for Identification of Hyperpartisan Publishers. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Osman Mutlu, Ozan Arkan Can, and Erenay Dayanik. 2019. Team Howard Beale at SemEval-2019 Task 4: Hyperpartisan News Detection with BERT. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Duc-Vu Nguyen, Thin Dang, and Ngan Nguyen. 2019. NLP@UIT at SemEval-2019 Task 4: The Paparazzo Hyperpartisan News Detector. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Zhiyuan Ning, Yuanzhen Lin, and Ruichao Zhong. 2019. Team Peter-Parker at SemEval-2019 Task 4: BERT-Based Method in Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Niko Palić, Juraj Vladika, Dominik Cubelić, Ivan Lovrencic, and Jan Snajder. 2019. TakeLab at SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Olga Papadopoulou, Giorgos Kordopatis-Zilos, Markos Zampoglou, Symeon Papadopoulos, and Yiannis Kompatsiaris. 2019. Brenda Starr at SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Martin Potthast, Tim Gollub, Matthias Hagen, and Benno Stein. 2018a. The Clickbait Challenge 2017: Towards a Regression Model for Clickbait Strength. *CoRR*, abs/1812.10847.

Martin Potthast, Tim Gollub, Kristof Komlossy, Sebastian Schuster, Matti Wiegmann, Erika Patricia Garces Fernandez, Matthias Hagen, and Benno Stein. 2018b. Crowdsourcing a Large Corpus of Clickbait on Twitter. In *27th International Conference on Computational Linguistics (COLING 2018)*, pages 1498–1507. The COLING 2018 Organizing Committee.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. TIRA Integrated Research Architecture. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018c. A Stylometric Inquiry into Hyperpartisan and Fake News. In *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 231–240. Association for Computational Linguistics.

Martin Potthast, Sebastian Köpsel, Benno Stein, and Matthias Hagen. 2016. Clickbait Detection. In

Advances in Information Retrieval. 38th European Conference on IR Research (ECIR 2016), volume 9626 of *Lecture Notes in Computer Science*, pages 810–817, Berlin Heidelberg New York. Springer.

Abdelrhman Saleh, Ramy Baly, Alberto Barrón-Cedeño, Giovanni Da San Martino, Mitra Mohtarami, Preslav Nakov, and James Glass. 2019. Team QCRI-MIT at SemEval-2019 Task 4: Propaganda Analysis Meets Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Saptarshi Sengupta and Ted Pedersen. 2019. Duluth at SemEval-2019 Task 4: The Pioquinto Manterola Hyperpartisan News Detector. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Daniel Shaprin, Giovanni Da San Martino, Alberto Barrón-Cedeño, and Preslav Nakov. 2019. Team Jack Ryder at SemEval-2019 Task 4: Using BERT Representations for Detecting Hyperpartisan News. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Vertika Srivastava, Ankita Gupta, Divya Prakash, Sudeep Sahoo, Rohit R. R., and Yeon Hyang Kim. 2019. Vernon-fenwick at SemEval-2019 Task 4: Hyperpartisan News Detection using Lexical and Semantic Features. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Bozhidar Stevanoski and Sonja Gievska. 2019. Team Ned Leeds at SemEval-2019 Task 4: Exploring Language Indicators of Hyperpartisan Reporting. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Emmanuel Vincent and Maria Mestre. 2018. Crowdsourced Measure of News Articles Bias: Assessing Contributors' Reliability. In *Proceedings of the 1st Workshop on Subjectivity, Ambiguity and Disagreement (SAD) in Crowdsourcing*, pages 1–10.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical Attention Networks for Document Classification. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Chia-Lun Yeh, Babak Loni, and Anne Schuth. 2019. Tom Jumbo-Grumbo at SemEval-2019 Task 4: Hyperpartisan News Detection with GloVe vectors and SVM. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Albin Zehe, Lena Hettinger, Stefan Ernst, Christian Hauptmann, and Andreas Hotho. 2019. Team Xenophilius Lovegood at SemEval-2019 Task 4: Hyperpartisanship Classification using Convolutional Neural Networks. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Chiyu Zhang, Arun Rajendran, and Muhammad Abdul-Mageed. 2019. UBC-NLP at SemEval-2019 Task 4: Hyperpartisan News Detection With Attention-Based Bi-LSTMs. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Team Bertha von Suttner at SemEval-2019 Task 4: Hyperpartisan News Detection using ELMo Sentence Representation Convolutional Network

Ye Jiang, Johann Petrak, Xingyi Song, Kalina Bontcheva, Diana Maynard
Department of Computer Science
University of Sheffield
Sheffield , UK
{yjiang18,johann.petrak,x.song,
k.bontcheva,d.maynard}@sheffield.ac.uk

Abstract

This paper describes the participation of team "bertha-von-suttner" in the SemEval2019 task 4 Hyperpartisan News Detection task. Our system[1] uses sentence representations from averaged word embeddings generated from the pre-trained ELMo model with Convolutional Neural Networks and Batch Normalization for predicting hyperpartisan news. The final predictions were generated from the averaged predictions of an ensemble of models. With this architecture, our system ranked in first place, based on accuracy, the official scoring metric.

1 Introduction

Hyperpartisan news is typically defined as news which exhibits an extremely biased opinion in favour of one side, or unreasoning allegiance to one party (Potthast et al., 2017). SemEval-2019 Task 4 on "Hyperpartisan News Detection" (Kiesel et al., 2019) is a document-level classification task which requires building a precise and reliable algorithm to automatically discriminate hyperpartisan news from more balanced stories.

One of the major challenges of this task is that the model must have the ability to adapt to a large range of article sizes. In one of the training data sets, the *by-publisher* corpus, the average article length is 796 tokens, but the longest document has 93,714 tokens. Most state-of-the-art neural network approaches for document classification use a token sequence as network input (Kim, 2014; Yin and Schütze, 2016; Conneau et al., 2016). This implies either a high computational cost when a very large maximum sequence length is used to fully represent the longest articles, or alternatively potentially a significant loss of information if the sequence length is restricted to a manageable number of initial tokens from the document.

In this paper, we introduce the **E**LMo **S**entence **R**epresentation **C**onvolutional (ESRC) Network. We first pre-calculate sentence level embeddings as the average of ELMo (Peters et al., 2018) word embeddings for each sentence, and represent the document as a sequence of such sentence embeddings. We then apply a lightweight convolutional Neural Network (CNN), along with Batch Normalization (BN), to learn the document representations and predict the hyperpartisan classification.

Two types of data set have been made available for the task. The *by-publisher* corpus contains 750K articles which were automatically classified based on a categorization of the political bias of the news source. This dataset was split into a training set of 600K articles and a validation set of 150K articles, where all the articles in the validation set originated from sources not in the training set. The second set, *by-article*, contains just 645 articles which were labelled manually. The final evaluation (Potthast et al., 2019) was carried out on a dataset of 628 articles which were also labelled manually.

We created several models based on the two datasets and evaluated them using cross-validation on the *by-article* training set (as the final test set was not available to the participants and it was only available for a maximum of three evaluations). In order to investigate the usefulness of the *by-publisher* training data for training a model that performs well on the manually annotated *by-article* corpus, we experimented with various kinds of pre-training and fine-tuning, and found that any kind of use of the *by-publisher* corpus was actually harmful and decreased the usefulness of the model. A CNN model which used ELMo-based sentence embeddings to represent the article, and was trained on the *by-article* set only,

[1]The code is available at
https://github.com/GateNLP/semeval2019-hyperpartisan-bertha-von-suttner

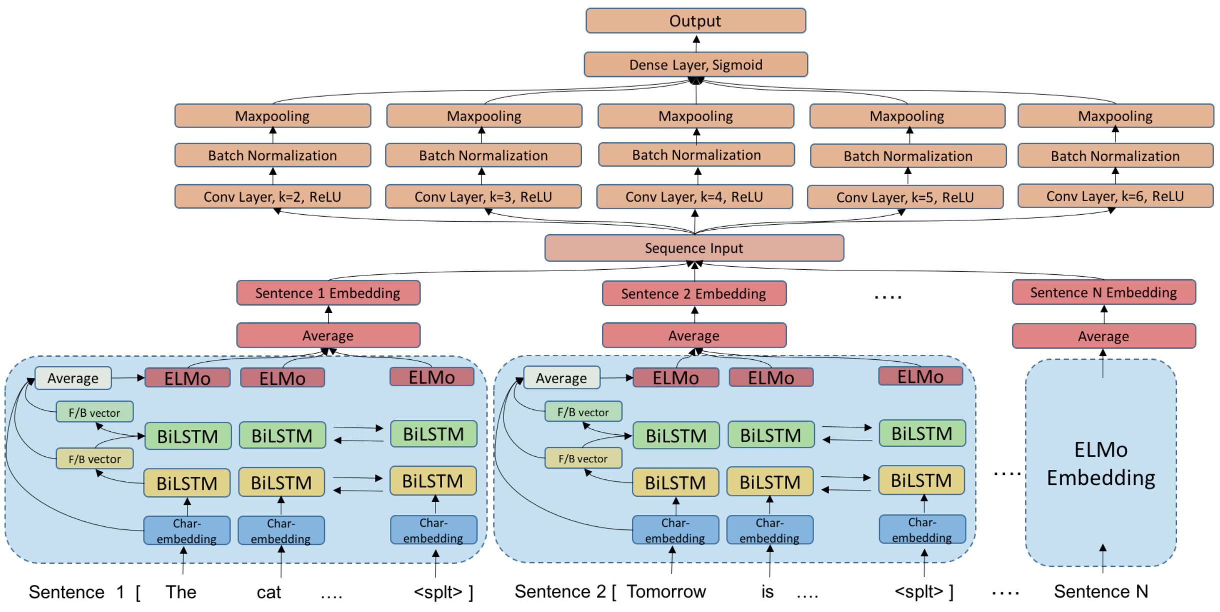


Figure 1: System architecture, *F/B vector* denotes Forward/Backward hidden state from BiLSTM layers.

turned out to outperform all other attempts.

2 System Description

In our model, we represent each article as a sequence of sentence embeddings, where each sentence embedding is calculated as the averaged word embeddings generated from a pre-trained ELMO model. The network consists of 5 parallel convolutional layers with kernel sizes 2,3,4,5,6 and 512 output features, each followed by a ReLU non-linearity, batch normalization, and max-pooling. All the results of the max-pooling layers are combined and go through a final fully connected layer with a sigmoid activation function for the final binary classification. Our model architecture is shown in Figure 1.

2.1 Data

The maximum, mean, and minimum numbers of tokens in the *by-article* corpus are: 6470, 666, 19 respectively, and in the *by-publisher* are: 93714, 796, 10 respectively. This makes it impractical to directly use word level representations as the input for our models. As a simple and easy to calculate compromise between representing the details of the article and as much of a longer article as possible, we represent the article as a sequence of sentence embeddings which are calculated as the average of the word embeddings of a sentence. This can be done using any pre-trained word embeddings and does not require a large training set for training or pre-training, so can be easily applied to even the small *by-article* corpus. To form the input sequence for our network, a maximum of the 200 initial tokens per sentence was used for each sentence embedding and a maximum of 200 sentences was used per article. The title of the article was used as the first article sentence for each document.

2.2 Preprocessing

Our model is character-based, which enabled us to only perform minimal pre-processing. We extract the title and article text from the original XML representation. All the original HTML paragraphs in the text cause a sentence break; the remaining text paragraphs have been split into sentences using `Spacy`. The original case of the text was maintained.

Whitespace is normalized to a single space between tokens; numbers are replaced by a special number token; and all punctuation and other special characters are preserved as input to the pre-trained ELMo model.

2.3 Deep Contextualized Word Representation

Traditionally, the input to CNNs is a set of pre-trained word vectors such as Word2Vec (Mikolov et al., 2013), Glove (Pennington et al., 2014), or Fasttext (Bojanowski et al., 2017). In our model, we use the AllenNLP library to generate ELMo embeddings, in which the word representation is learned from character-based units as well as contextual information from the news articles. These character-based word representations allow our model to pick up on morphological features that word-level embeddings could miss, and a valid word representation can be formed even for out-of-vocabulary words. Furthermore, ELMo uses two bi-directional LSTM (Gers et al., 1999) layers to learn the contextual information from the text, which makes it capable of disambiguating the same word into different representations based on its context.

We use the original[2] pre-trained ELMo model to output three vectors for each word. Each vector corresponds to a layer output from the ELMo pre-trained model. Then, we take the average of all three vectors to form the final word vector, and compute the sentence vector by averaging the word vectors in the sentence.

2.4 Convolutional Layers

We combine 5 convolutional layers for different kernel sizes. Each layer is then followed by a non-linear activation function ReLU.

2.5 Batch Normalization

Batch Normalization (BN) is a method for reducing internal covariate shift in neural networks (Ioffe and Szegedy, 2015). BN normalizes the input distribution by subtracting the batch mean and dividing by the batch standard deviation, so that the ranges of input distribution between each layer stay the same. This allows the model to have a higher learning rate, so that the training speed is accelerated. It also reduces overfitting by decreasing the dependence of weight initialization between each layer. The original paper suggested that BN should be applied before the activation layer, but we apply it after the activation layer, after observing better performance in our model this way round. We also applied weighted moving-mean and moving-variance to avoid updating the mean and variance so aggressively in the mini-batch during training time.

2.6 Fully Connected Layer

We perform max-pooling on the output of the batch-normalization layers. Then the outputs of the max-pooling for all convolution layers are combined to form the input to a fully connected layer, which maps to a single output, followed by the Sigmoid function for the binary classification task.

[2] `elmo_2x4096_512_2048cnn_2xhighway`

3 Experiments and Results

The generated ELMo embedding contains three vectors for each word, where each vector corresponds to one of the output layers from the pre-trained model. We average the three vectors to generate word representations which contain morphological and contextual information, and compute the sentence vectors by averaging all the word vectors in each sentence. We take a maximum of 200 words for each sentence and a maximum of 200 sentences for each article. If a document has fewer than 200 sentences, we pad the number of sentences out to 200.

Our models are built by using the Keras library with a Tensorflow backend. All the results are shown in Table 1. The table shows for each model the accuracy obtained on the *by-article* training set, and for the submitted models, the *by-publisher* test set, and the hidden *by-article* test set (which unlike the other two, was not available to participants).

In order to investigate the correlation between the two datasets, we first built the `ESRC-publisher` model which is trained on a randomly selected 100K out of the 750K articles from the *by-publisher* corpus, as it is impractical to generate ELMo embeddings for the entire corpus. We also fine-tuned the `ESRC-publisher` model based on the *by-article* set to obtain the `ESRC-publisher-article` model by freezing the weights of all but the last layer of the model. Finally we trained the `ESRC-article` model only on the *by-article* set, one version without and one version (`ESRC-article-BN`) with the additional batch normalization (BN) layer. The accuracy for the `ERC-publisher` model is from evaluating on the whole `by-article` training set, while all other evaluations on the `by-article` training set were carried out using a 10-fold cross validation. However, because of the very limited size of that corpus, the evaluation part of each fold was also used for early stopping and model selection within each fold.

For the evaluation on the hidden test set, we selected the best three models from the 10-folds, according to the accuracy on the evaluation set of each fold to form an averaged ensemble model, `ESRC-article-BN-Ens`.

For comparison, the table also shows the results for an earlier version of the model, `GloVe-article`, which used GloVe word embeddings (6 billion words, 300 dimensional) to represent up to the first 400 words of the article and did not use batch normalization.

Models	**By-Article Training**
GloVe-article	0.7963
ESRC-publisher	0.5643
ESRC-publisher-article	0.8189
ESRC-article	0.8182
ESRC-article-BN	0.8387
ESRC-article-BN-Ens	**0.8404**
Submitted Models	**By-Article Test**
GloVe-article	0.7659
ESRC-article-BN-Ens	0.8216
Submitted Models	**By-Publisher Test**
GloVe-article	0.6435
ESRC-article-BN-Ens	0.5947

Table 1: System comparison (accuracy).

The parameters in our models are as follows: we used 5 convolutional layers with kernel sizes ($k = 2, 3, 4, 5, 6$) and 512 output features. The momentum in the batch normalization is set to 0.7.[3] We used the default Adam algorithm as the optimizer, and Binary Cross-Entropy as the loss function. The batch size was set to 32 and the fixed number of epochs used was 30. The final best model after 30 epochs was used.

4 Discussion and Conclusion

The `ESRC-publisher` model performs extremely badly on the *by-article* evaluation data. Even fine-tuning the `ESRC-publisher` model on the *by-article* corpus produces models which perform worse than a model that is trained only on the *by-article* data. This confirms results from earlier experiments with simpler models that any use of the *by-publisher* data only hurts the model. We assume that the algorithm used for assigning the labels to this dataset just does not reflect any information about hyperpartisan articles sufficiently to be helpful. For this reason, the `GloVe-article` model also outperforms the ESRC-article-BN-Ens model on the by-publisher dataset.

A quick manual inspection of the data showed that the source of an article is insufficient by far to identify articles as hyperpartisan or not. It would be interesting to know how the algorithm used for creating the *by-publisher* corpus actually performs on the *by-article* corpus. To get maximum performance on the *by-article* dataset, we therefore decided to completely ignore the *by-publisher* data for our final model. The use of BN also showed significant improvement.

Since we use a CNN with a comparatively large number of parameters in relation to the size of the training set which is rather small, we expect significant variance in the generated models and therefore use the average of an ensemble of several models for the final predictions.

[3]This was determined by exploring values from 0.1 to 0.9 at an earlier stage of the experiments and kept, so it may not be the optimal value.

5 Acknowledgements

Research partially supported by a Grantham Centre for Sustainable Future Scholarship, a Google Faculty Research Award 2017, and projects funded by the European Commissions Horizon 2020 research and innovation programme under grant agreements No. 654024 SoBigData and No. 825297 WeVerify.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with lstm.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. TIRA Integrated Research Architecture. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2017. A stylometric inquiry into hyperpartisan and fake news. *arXiv preprint arXiv:1702.05638*.

Wenpeng Yin and Hinrich Schütze. 2016. Multichannel variable-size convolution for sentence classification. *arXiv preprint arXiv:1603.04513*.

RumourEval 2019: Determining Rumour Veracity and Support for Rumours

Genevieve Gorrell[1] and **Elena Kochkina**[3,4] and **Maria Liakata**[3,4] and **Ahmet Aker**[1] and **Arkaitz Zubiaga**[5] and **Kalina Bontcheva**[1] and **Leon Derczynski**[2]

[1]University of Sheffield, UK
(g.gorrell,a.aker,k.bontcheva)@sheffield.ac.uk

[2]IT University of Copenhagen, Denmark
ld@itu.dk

[3]University of Warwick, UK
(e.kochkina,m.liakata)@warwick.ac.uk
[4]Alan Turing Institute, UK

[5]Queen Mary University of London, UK
a.zubiaga@qmul.ac.uk

Abstract

Since the first RumourEval shared task in 2017, interest in automated claim validation has greatly increased, as the danger of "fake news" has become a mainstream concern. However automated support for rumour verification remains in its infancy. It is therefore important that a shared task in this area continues to provide a focus for effort, which is likely to increase. Rumour verification is characterised by the need to consider evolving conversations and news updates to reach a verdict on a rumour's veracity. As in RumourEval 2017 we provided a dataset of dubious posts and ensuing conversations in social media, annotated both for stance and veracity. The social media rumours stem from a variety of breaking news stories and the dataset is expanded to include Reddit as well as new Twitter posts. There were two concrete tasks; rumour stance prediction and rumour verification, which we present in detail along with results achieved by participants. We received 22 system submissions (a 70% increase from RumourEval 2017) many of which used state-of-the-art methodology to tackle the challenges involved.

1 Introduction

1.1 Background

Since the first RumourEval shared task in 2017 (Derczynski et al., 2017), interest in automated verification of rumours has deepened, as research has demonstrated the potential impact of false claims on important political outcomes (Allcott and Gentzkow, 2017). Living in a "post-truth world", in which perceived truth can matter more than actual truth (Dale, 2017), the dangers posed by unchecked market forces and cheap platforms, as well as poor ability by many readers to discern credible information, are evident. As a result the importance of educating young people about critical thinking is increasingly emphasised.[1]. Moreover the European Commission's High Level Expert Group on Fake News provides tools to empower users and journalists to tackle disinformation as one of the five pillars of their recommended approach.[2] Platforms are increasingly motivated to engage with the problem of damaging content that appears on them, as society moves toward a consensus regarding their level of responsibility. Independent fact checking efforts, such as Snopes[3], Full Fact[4], Chequeado[5], are also becoming valued resources (Konstantinovskiy et al., 2018). Zubiaga et al. (2018) present an extensive list of projects. Effort so far is often manual, and struggles to keep up with the large volume of online material.

Within NLP research the tasks of stance classification of news articles and social media posts and the creation of systems to automatically identify false content are gaining momentum. Work in credibility assessment has been around since 2011 (Castillo et al., 2011), making use initially

[1]http://www.bbc.co.uk/mediacentre/latestnews/2017/fake-news
[2]http://ec.europa.eu/newsroom/dae/document.cfm?doc_id=50271
[3]https://www.snopes.com/
[4]https://fullfact.org/
[5]http://chequeado.com/

Veracity prediction. Example 1:

u1: Hostage-taker in supermarket siege killed, reports say. #ParisAttacks LINK [true]

Veracity prediction. Example 2:

u1: OMG. #Prince rumoured to be performing in Toronto today. Exciting! [false]

Table 1: Examples of source tweets with veracity value

of local features. Fact checking is a broad complex task, challenging the resourcefulness of even a human expert. Claims such as ”we send the EU 350 million a week” which is partially true would need to be decomposed into statements to be checked against knowledge bases and multiple sources. Ways of automating fact checking has inspired researchers (Vlachos and Riedel, 2015) and has resulted in a new shared task FEVER.[6] Other research has focused on stylistic tells of untrustworthiness in the source itself (Conroy et al., 2015; Singhania et al., 2017). Rumour verification is a particular case of fact checking. Rumours are *“circulating stories of questionable veracity, which are apparently credible but hard to verify, and produce sufficient skepticism and/or anxiety so as to motivate finding out the actual truth”* (Zubiaga et al., 2016). One can distinguish several component to a rumour resolution pipeline such as rumour detection, rumour tracking and stance classification, leading to the final outcome of determining the veracity of a rumour (Zubiaga et al., 2018). Thus what characterises rumour verification compared to other types of fact checking is time sensitivity and the importance of dynamic interactions between users, their stance and information propagation. Initial work on rumour detection and stance classification (Qazvinian et al., 2011) was succeeded by more elaborate systems and annotation schemas (Kumar and Geethakumari, 2014; Zhang et al., 2015; Shao et al., 2016; Zubiaga et al., 2016). Vosoughi (2015) demonstrated the value of making use of propagation information, i.e. the ensuing discussion, in rumour verification. Stance detection is the task of classifying a text according to the position it takes with respect to a statement. Research supports the importance of this subtask as a first step to veracity identification. (Ferreira and Vlachos, 2016; Enayet and El-Beltagy, 2017). Crowd response, stance and the details of rumour propagation feature in the work by Chen et al. (2016) as well as the most successful system in RumourEval 2017 (Enayet and El-Beltagy, 2017), and the highest performing systems in RumourEval 2019.

1.2 Datasets for rumour verification

The UK fact-checking charity Full Fact provides a roadmap[7] for development of automated fact checking. They cite open and shared evaluation as one of their five principles for international collaboration, demonstrating the continuing relevance of shared tasks in this area. Shared datasets are a crucial part of the joint endeavour. Datasets for rumour resolution are still relatively few, and likely to be in increasing demand. In addition to the data from RumourEval 2017, the dataset released by Kwon et al. (2017) is also suitable for veracity classification. It includes 51 true rumours and 60 false rumours, where each rumour includes a stream of tweets associated with it. Twitter 15 and 16 datasets (Ma et al., 2018) contain claim propagation trees and combine tasks of rumour detection and verification in one four-way classification task (Non-rumour, True, False, Unverified). A Sina Weibo corpus is also available (Wu et al., 2015), in which 5000 posts are classified for veracity, but responses are not available. Partially generated statistical claim checking data is now becoming available in the context of the FEVER shared task, mentioned above, but is not suitable for this type of work. Twitter continues to be a highly relevant platform for rumour verification, being popular with the public as well as politicians. RumourEval 2019 also includes Reddit

[6]https://sheffieldnlp.github.io/fever/

[7]https://fullfact.org/media/uploads/full_fact-the_state_of_automated_factchecking_aug_2016.pdf

data, thus providing more diversity in the types of users, more focussed discussions and longer texts.

1.3 RumourEval 2017 vs 2019

RumourEval 2019 furthers progress on stance detection and rumour verification, both still unbested NLP tasks. They are currently moderately well performed for English short texts (tweets), with data existing in a few other languages (notably as part of IberEval). In 2019, many more teams took part, demonstrating the rising relevance of the tasks. Specifically, as in 2017, RumourEval 2019 comprises two subtasks:

- In subtask A, given a source tweet, tweets in a conversation thread discussing the claim are classified as either supporting, denying, querying or commenting on the rumour mentioned by the source tweet
- In subtask B, the rumour introduced by the source tweet that spawned the discussion is classified as true, false or unverified.

In 2017 we had two variants of the task, a closed and an open one.

- In the open variant, a system could consider the source tweet itself, the discussion as well as additional background information.
- In the closed variant, only the source tweet and the ensuing discussion were used by systems.

Eight teams entered subtask A, achieving accuracies ranging from 0.635 to 0.784. In the open variant of subtask B, only one team participated, gaining an accuracy of 0.393 and demonstrating that the addition of a feature for the presence of the rumour in the supplied additional materials does improve their score. Five teams entered the closed variant of task B, scoring between 0.286 and 0.536. Only one of these made use of the discussion material, specifically the percentage of responses querying, denying and supporting the rumour but scored joint highest on accuracy and achieved the lowest RMSE. A variety of machine learning algorithms were employed. Among traditional approaches, a gradient boosting classifier achieved the second best score in task A, and a support vector machine achieved a fair score in task A and first place in task B. However, deep learning approaches also fared well; an LSTM-based approach took first place in task A and an approach using CNN took second place in task B, though performing less well in task A. Other teams used different kinds of ensembles and cascades of traditional and deep learning supervised approaches.

For 2019 we wanted to encourage participants to be more innovative in the information they make use of, particularly in exploiting the output of task A in their task B approaches.

We extended the challenges through the addition of new data and by including Reddit posts.

In order to encourage more information-rich approaches, we combined variants of subtask B into a single task, allowing participants to use additional material. This was selected to provide a range of options whilst being temporally appropriate to the rumours in order to mimic the conditions of a real world rumour checking scenario.

1.4 Subtask A - SDQC support classification

Related to the objective of predicting a rumour's veracity, and as a first step in a rumour verification pipeline, Subtask A deals with the complementary objective of tracking how other sources orient to the accuracy of the rumourous story. A key step in the analysis of the surrounding discourse is to determine how other users in social media regard the rumour (Procter et al., 2013). Given a source post containing a rumourous claim and a conversation thread discussing the rumour as input, the objective is to label each of the posts in the conversation thread with respect to their stance towards the rumour.

Success on this task supports success on task B by providing additional context and information; for example, where the discussion ends in a number of agreements, it could be inferred that human respondents have verified the rumour. In this way, task A provides an intermediate challenge in which a larger number of data points can be provided. See Table 2 for an example conversation thread and refer to Derczynski et al. (2017) for more details about the task definition.

1.5 Subtask B - Veracity prediction

As in RumourEval 2017 (Derczynski et al., 2017), the goal of subtask B is to predict the veracity of a given rumour, where the latter is presented in the form of a post reporting an update associated with a newsworthy event. Given such a claim as input,

SDQC support classification. Example 1:

u1: We understand that there are two gunmen and up to a dozen hostages inside the cafe under siege at Sydney.. ISIS flags remain on display #7News **[support]**
 u2: @u1 not ISIS flags **[deny]**
 u3: @u1 sorry - how do you know its an ISIS flag? Can you actually confirm that? **[query]**
  u4: @u3 no she cant cos its actually not **[deny]**
 u5: @u1 More on situation at Martin Place in Sydney, AU LINK **[comment]**
 u6: @u1 Have you actually confirmed its an ISIS flag or are you talking shit **[query]**

SDQC support classification. Example 2:

u1: These are not timid colours; soldiers back guarding Tomb of Unknown Soldier after today's shooting #StandforCanada PICTURE **[support]**
 u2: @u1 Apparently a hoax. Best to take Tweet down. [deny]
 u3: @u1 This photo was taken this morning, before the shooting. [deny]
 u4: @u1 I dont believe there are soldiers guarding this area right now. [deny]
  u5: @u4 wondered as well. Ive reached out to someone who would know just to confirm that. Hopefully get response soon. [comment]
   u4: @u5 ok, thanks. [comment]

Table 2: Examples of tree-structured threads discussing the veracity of a rumour, where the label associated with each tweet is the target of the SDQC support classification task.

plus additional data such as stance data classified in task A and any other information teams chose to use from the selection provided, systems return a label describing the anticipated veracity of the rumour. Examples are given in Table 1. In addition to returning a classification of true, or false, a confidence score was also required, allowing for a finer grained evaluation. A confidence score of 0 should be returned if the rumour is unverified.

2 Data & Resources- RumourEval 2019

The data are structured as follows. Source posts introduce a rumour, and may be true, false or unverified. These are accompanied by an ensuing discussion (tree-shaped) in which users support, deny, comment or query (SDCQ) the rumour in the source text. This is illustrated in figure 1 with an example rumour about Putin. Note that source posts also need to be annotated for stance, as the way a post presents a rumour usually gives stance information also. For example, when introducing a rumour, an implicit "support" stance may be present, in that the rumour is assumed to convey valid information. In the Reddit data, rumours were often introduced with an implicit "query", as they were presented for discussion/debunking.

The RumourEval 2017 corpus contains 297 source tweets grouped into eight breaking news events, and a total of 7100 discussion tweets. This became training data in 2019, and was augmented with new Twitter test data and new Reddit material. The Reddit material was split into training and test sets. Each are discussed in turn below.

In RumourEval 2017 along with the tweet threads, we also provided additional context that participants could make use of (Derczynski et al., 2017). However, only one system had made use of this additional context. Due to lack of time such context data was not provided in RumourEval 2019 but we would look into re-introducing this in future editions of the task.

2.1 English Twitter data about natural disasters

The additional English Twitter testing data is about natural disasters. In such events, where chaos dominates the situation, rumours are spread on various issues and false rumours have the potential to increase the chaos. Detecting such false rumours are important to plan actions that will

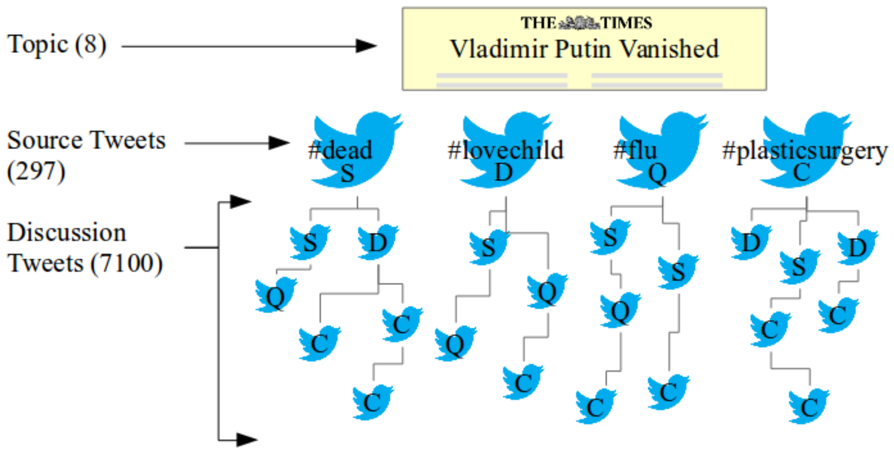


Figure 1: Structure of the first rumours corpus

eliminate the additional negative impact on the already existing chaotic situation. Therefore, for this year we decided to introduce such a dataset as test data. To collect this dataset rumours about natural disasters were chosen manually through Snopes.com and Politifact.com: we searched manually for rumours about known natural disasters such as hurricanes, floods, etc. If the search returned some results, we quickly scanned this result list for social media posts (specifically tweets) that people had created about the disaster and which had been verified by the debunking web-site.

Once we collected the rumour introducing tweets (the source tweets) we aimed to collect also the cascades, i.e. the reactions/replies to the source tweet. The replies encode the reactions (stance information) of other users to the rumour and can be of importance when verifying the rumour. To collect the replies we used an existing scraper (Zubiaga et al., 2016). The number of source tweets of different veracities and replies of different stances are given in Tables 3 and 4.[8]

2.1.1 Annotation of new English Twitter data

As noted above a rumour consists of a source tweet and a thread of tweets that respond to the source one, where the source tweet contains the rumour. The veracity of each source tweet is already known a priori. However, the dataset is missing stance labels for the replies. To get also the stance labels we performed annotation through crowd sourcing. Zubiaga et al. (2016) distinguish between the following stance labels for each replying tweet: supporting, denying, questioning and commenting.

Following the same strategies and design reported by Zubiaga et al. (2016) we posted our datasets for stance annotation to FigureEight (F8)[9]. We applied a restriction so that annotation could be performed only by people from the USA and UK. We also made sure that each annotation was performed maximum by 10 annotators and that an annotator agreement of min. 70% was met. Note if the agreement of 70% was met with fewer annotators then the system would not force an annotation to be done by 10 annotators but would finish earlier. The system requires 10 annotators if the minimum agreement requirement is not met. Each annotator saw five source tweets on a page. The source tweets were accompanied by replying tweets followed by the stance labels to choose from. Each page showed also instructions and definitions about the stance labels. We paid for each tweet annotation 3 US Dollar Cents.

The agreement among the annotators is directly taken from F8s aggregated scores and is computed based on percentage agreement. On the entire dataset we have 76.2% agreement.

We also computed the distribution of stances provided for the replying tweets (see Tables 3 and 4). As we see from the tables, overall the distribution of stances is skewed towards the comment category. This is also the case with the PHEME dataset reported by Zubiaga et al. (2016).

[8]The labels were taken from the debunking web-sites. As in the RumourEval2017 test data the false rumours dominate. However, unlike the previous dataset the number of unverified rumours is proportionally smaller compared to the other two classes. In the 2017 dataset the test data included 12 false rumours, 8 true and 8 unverified ones.

[9]www.figure-eight.com

2.2 Reddit Data

Rumours were identified on Reddit by manually searching debunking forums and current affairs forums to identify suitable threads. Reddit discussions are deeper than Twitter discussions, with often a complex conversational structure exploring the topic. They are usually introduced by a post implicitly querying the rumour, unlike Twitter rumours which are more often presented as valid information and therefore the source tweets usually support the rumour. The Reddit material is less time-sensitive than the Twitter material, and may discuss long-standing conspiracy theories, for example. Threads were downloaded using a bespoke script.

2.2.1 Annotation of Reddit discussions

Since the Reddit discussions are complex, there is more of a danger that careless annotators won't distinguish between posts that disagree with the immediately preceding comment and posts that disagree with the rumour. A response such as "absolutely!" might therefore get a high agreement from annotators who all made the mistake of annotating it as "support", even if it was in response to a preceding comment which denied the rumour. To avoid this, an extensive quiz of 51 test questions was used to ensure that annotators understood the task properly. Reddit threads tend to be longer and more diverse, leading to a more challenging task as discussion may be only loosely related to the main topic, leading to a preponderance of "comments" (88% overall compared with 67% in the Twitter data). Tables 3 and 4 give totals in training and test data for both tasks alongside the figures for Twitter data.

Up to five judgements were collected, or an agreement of 0.7, whichever came first. Since Reddit annotators were highly trained by the time they were accepted on the task, this was found sufficient. Four US dollar cents per post was offered, which is higher than usual for a Figure Eight task, in order to attract annotators to this relatively hard task. The final macro-agreement for the entire Reddit set is 78%, and an average of 3.84 annotations annotated each item. For "support" items, more annotations were required, at 4.22 on average, and a lower macro agreement was achieved of 67%. Similarly for deny items, 4.04 judgements were obtained on average and a macro agreement of 63% was achieved. For query items, 4.36 judgements on average were obtained and a macro agreement of 64% was achieved.

	Supp.	**Deny**	**Query**	**Com.**	**Total**
Twitter Train	1004	415	464	3685	5568
Reddit Train	23	45	51	1015	1134
Total Train	1027	460	515	4700	6702
Twitter Test	141	92	62	771	1066
Reddit Test	16	54	31	705	806
Total Test	157	146	93	1476	1872
Total Task A	1184	606	608	6176	8574

Table 3: Task A corpus

	True	**False**	**Unver.**	**Total**
Twitter Train	145	74	106	325
Reddit Train	9	24	7	40
Total Train	154	98	113	365
Twitter Test	22	30	4	56
Reddit Test	9	10	6	25
Total Test	31	40	10	81
Total Task B	185	138	123	446

Table 4: Task B corpus

For Task B, rumours were annotated for veracity with the aid of Snopes and similar sites. This is a change from RumourEval 2017, where manually-annotated veracity was assigned. Instead, we used community experts working professionally in a range of organisations to construct the Task B veracity judgments. The volume of data was also significantly extended beyond e.g. the 21 stories in the test set of RumourEval 2017 Task B.

3 Evaluation

In task A, stance classification, care must be taken to accommodate the skew towards the "comment" class, which dominates, as well as being the least helpful type of data in establishing rumour veracity. Therefore we used macro-averaged F1 to evaluate performance on task A.

In task B participants supply a true/false classification for each rumour, as well as a confidence score. Macro-averaged F1 was again the score of choice to evaluate the overall classification. For the confidence score, a root mean squared error (RMSE, a popular metric that differs only from the Brier score in being its square root) was calculated relative to a reference confidence of 1. Unverified rumours were considered correctly annotated if they received a confidence score of zero regardless of true/false classification.

The previous RumourEval task used accuracy as the evaluation metric, but that approach allowed higher scores to be obtained through less sensitivity to minority classes. For the stance task, 80% of test items were comments, and this is the least interesting class. For the verification task, class imbalance is not so extreme, with 50%"false" in the dataset and close to 40% "true" (the remainder are "unverified").

Whilst participants weren't evaluated on accuracy for task A, we note that generally speaking, teams that obtained higher macro F1 scores also obtained higher accuracies, and that around 50% of the teams obtained accuracies higher than might be obtained simply by assigning all items to the comment class (majority baseline). However, the correlation between accuracy and macro F1 was only 0.47, and use of macro F1 revealed that three teams surged ahead. For task B, where class imbalance was less pronounced, the relationship between accuracy and macro F1 was much closer, with a correlation of 0.87, though again, F1 was the better differentiator. Interestingly, RMSE showed a stronger relationship with macro F1 than with accuracy (correlations -0.92 vs -0.77).

User or Team name	Subtask B, MacroF	Subtask B, RMSE	Subtask A, Macro F
eventAI	**0.5765 (1)**	**0.6078 (1)**	0.5776 (3)
WeST (CLEARumor)	0.2856 (2)	0.7642 (2)	0.3740 (11)
GWU NLP LAB	0.2620 (3)	0.8012 (3)	0.4352 (7)
BLCU NLP	0.2525 (4)	0.8179 (5)	**0.6187 (1)**
shaheyu	0.2284 (5)	0.8081 (4)	0.3053 (17)
Columbia	0.2244 (6)	0.8623 (7)	0.3625 (13)
mukundyr	0.2244 (6)	0.8623 (7)	0.3404 (15)
Xinthl	0.2238 (7)	0.8623 (7)	0.2297 (18)
lzr	0.2238 (7)	0.8678 (8)	0.3404 (15)
UPV-28-UNITO	0.1996 (8)	0.8264 (6)	0.4895 (4)
NimbusTwoThousand	0.0950 (9)	0.9148 (9)	0.1272 (19)
nx1 (deanjjones)	-	-	0.3267 (16)
jurebb	-	-	0.3537 (14)
UI-AI	-	-	0.3875 (10)
LECS	-	-	0.4384 (6)
magc	-	-	0.3927 (9)
BUT-FIT	-	-	0.6067 (2)
HLT(HITSZ)	-	-	0.4792 (5)
wshuyi	-	-	0.3699 (12)
SINAI-DL	-	-	0.4298 (8)
FINKI NLP 2018/2019 (late)	0.3326	0.6846	0.2165
IASBS (late)	0.1845	0.7857	0.2530
baseline branchLSTM	0.3364	0.7806	0.4929
baseline NileTMRG	0.3089	0.7698	-
baseline Majority class	0.2241	0.7115	0.2234

Table 5: Results table. Ranking is in brackets.

4 Baselines

We provided participants with our implementation of several baseline systems[10], described below.

4.1 Stance classification baseline

For subtask A we released a Keras (Chollet et al., 2015) implementation of branchLSTM, the winning system of RumourEval 2017 Task A (Kochkina et al., 2017). This system uses the conversation structure by splitting it into linear branches. It is a neural network architecture that uses LSTM layer(s) to process sequences of tweets, outputting a stance label at each time step. Each tweet is represented by the average of its word vectors [11] concatenated with a number of extra features. This baseline was outperformed by 3 submitted systems (BLCU NLP, BUT-FIT, eventAI).

4.2 Veracity classification baselines

For subtask B we provided two baselines.

1. A model which is an extension of branchLSTM (Kochkina et al., 2018) uses the same features as the stance classification system but produces a single output per branch. The veracity prediction for the thread is then decided using majority voting over per-branch outcomes.

2. The NileTMRG baseline (Enayet and El-Beltagy, 2017) is a linear SVM that uses a bag-of-words representation of the source tweet, concatenated features defined by the presence of URL, presence of hashtag and proportion of supporting, denying and querying tweets in the thread. In our implementation of NileTMRG e use the branchLSTM model to obtain stance labels for the tweets in the testing set rather than the model originally used in (Enayet and El-Beltagy, 2017).

Baseline systems in subtask B were outperformed by the winning system eventAI (outperforms both baselines) and a late submission by FINKI NLP (outperforms NileTMRG and reaches similar result to branchLSTM, see Table 5). If participants made their own run of the baseline sys-

[10]https://github.com/kochkinaelena/RumourEval2019

[11]We are using word2vec (Mikolov et al., 2013) model pretrained on the GoogleNews dataset (300d)

tems, their outcome might differ from ours due to variation in random seeds, package versions and hardware used.

5 Participant Systems and Results

We have had 22 system submissions at RumourEval 2019 (70% up from RumourEval 2017), confirming the significant increase in interest in this area. All submissions tackled subtask A (Rumour SDQC) and 13 systems attempted both tasks (more than a 100% increase). The participating systems and the results achieved can be found in Table 5. Note that system ranking is presented according to macro-F1 score in subtask B, which is considered the core task and the more challenging of the two. As in RumourEval 2017 subtask A was the more popular task of the two and whilst participation in both tasks has significantly increased, it is still the case that systems seem to focus and do better in one of the two tasks. Specifically, the best performing system in substask B (eventAI) ranked third in subtask A and the best performing system in subtask A (BLCU NLP) ranked fourth in subtask B. Three systems outperformed the branchLSTM subtask A baseline (BLCU NLP, BUT-FIT, eventAI), whereas almost all systems outperformed the majority baseline macro-F1 in this task. In subtask B, over 60% of systems outperformed the majority baseline in macro-F1, two systems outperformed the NILETMRG baseline (eventAI,FINKI-NLP–late) and one system (eventAI) beat both the NILETMRG and branchLSTM baselines.

The trend for neural approaches has demonstrably increased with almost all systems adopting a neural network (NN) architecture for their models, with the exception of the best performing system in subtask B (eventAI), which implemented an ensemble of classifiers (SVM,RF,LR), including a NN with three connected layers, where individual post representations are created using an LSTM with attention. This also considered a range of other features and postprocessing module to find similarities between source tweets. A similar ensemble model also considering sophisticated features and feature selection using RF would have ranked second in this task (FINKI-NLP, submitted late) as it outperformed the NILETMRG baseline. The second best performing system in subtask A (BUT-FIT) uses an ensemble of BERT (Devlin et al., 2018) models, which allows the pre-training of bidirectional representations to provide additional context. They experiment with different parameter settings and if the model increased overall performance it was added to the classifier. Interestingly the best performing system in task A (BLCU-NLP) and the third best (CLEARumor) also use pre-trained contextual embedding representations with BLCU-NLP using OpenAI GPT (Radford et al., 2018) and ClEARumor using ELMo (Peters et al., 2018). While most systems use single tweets or pairs of tweets (source-response) as their underlying structure to operate on, BLCU-NLP employ an inference chain-based system for this paper. Thus they consider the conversation thread starting with a source tweet, followed by replies, in which each one responds to an earlier one in time sequence. They take each conversation thread as an inference chain and concentrate on utilizing it to solve the problem of class imbalance in subtask A and training data scarcity in subtask B. They also have augmented the training data with external public datasets. Other popular neural models among participants include BiLSTM and LSTM. Judging from the approaches of two best performing systems in each of subtask A and B (BLCU-NLP and eventAI respectively) one could infer that: (1) for subtask A considering the sequence of earlier posts is important to identifying correctly the stance of a post towards the rumour (2) for rumour verification it is more important to consider a variety of different features.

6 Conclusion

We evaluated multiple teams in the tasks of rumour stance detection and rumour veracity evaluation. Interest in these tasks continues to increase, driving performance of systems higher and pushing the sophistication of systems, which are now often using state-of-the-art neural network methods and beyond. Further challenges include use of the rich context available, in terms of both time, conversation, and broader discourse during the evolution of rumours. Additionally, we need to work better with other languages. While we tried to make more available in this task, framing the task and annotating the data proved challenging and demanding. On the other hand, leaving stance detection just to English leaves the majority of the world without this important technology.

Acknowledgements

This work is supported by the European Commissions Horizon 2020 research and innovation programme under grant agreements No. 654024 SoBigData and No. 687847 COMRADES. This work was also partially supported by The Alan Turing Institute under the EPSRC grant EP/N510129/1. Cloud computing resources were kindly provided through a Microsoft Azure for Research Award. Work by Elena Kochkina was partially supported by the Leverhulme Trust through the Bridges Programme and Warwick CDT for Urban Science & Progress under the EPSRC Grant Number EP/L016400/1.

References

Hunt Allcott and Matthew Gentzkow. 2017. Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, 31(2):211–36.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684. ACM.

Weiling Chen, Chai Kiat Yeo, Chiew Tong Lau, and Bu Sung Lee. 2016. Behavior deviation: An anomaly detection view of rumour preemption. In *Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2016 IEEE 7th Annual*, pages 1–7. IEEE.

François Chollet et al. 2015. Keras. `https://keras.io`.

Niall J Conroy, Victoria L Rubin, and Yimin Chen. 2015. Automatic deception detection: Methods for finding fake news. *Proceedings of the Association for Information Science and Technology*, 52(1):1–4.

Robert Dale. 2017. Nlp in a post-truth world. *Natural Language Engineering*, 23(2):319–324.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. Semeval-2017 task 8: Rumoureval: Determining rumour veracity and support for rumours. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 69–76.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Omar Enayet and Samhaa R El-Beltagy. 2017. Niletmrg at semeval-2017 task 8: Determining rumour and veracity support for rumours on twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 470–474.

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 1163–1168.

Elena Kochkina, Maria Liakata, and Isabelle Augenstein. 2017. Turing at semeval-2017 task 8: Sequential approach to rumour stance classification with branch-lstm. *In Proceedings of SemEval.ACL.*

Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2018. All-in-one: Multi-task learning for rumour verification. *arXiv preprint arXiv:1806.03713*.

Lev Konstantinovskiy, Oliver Price, Mevan Babakar, and Arkaitz Zubiaga. 2018. Towards automated factchecking: Developing an annotation schema and benchmark for consistent automated claim detection. *arXiv preprint arXiv:1809.08193*.

KP Krishna Kumar and G Geethakumari. 2014. Detecting misinformation in online social networks using cognitive psychology. *Human-centric Computing and Information Sciences*, 4(1):14.

Sejeong Kwon, Meeyoung Cha, and Kyomin Jung. 2017. Rumor detection over varying time windows. *PloS one*, 12(1):e0168344.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2018. Rumor detection on twitter with tree-structured recursive neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1980–1989.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Rob Procter, Farida Vis, and Alex Voss. 2013. Reading the riots on twitter: methodological innovation for the analysis of big data. *International journal of social research methodology*, 16(3):197–214.

Vahed Qazvinian, Emily Rosengren, Dragomir R Radev, and Qiaozhu Mei. 2011. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1589–1599. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Time Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, Technical report, OpenAI.

Chengcheng Shao, Giovanni Luca Ciampaglia, Alessandro Flammini, and Filippo Menczer. 2016. Hoaxy: A platform for tracking online misinformation. In *Proceedings of the 25th international conference companion on world wide web*, pages 745–750. International World Wide Web Conferences Steering Committee.

Sneha Singhania, Nigel Fernandez, and Shrisha Rao. 2017. 3han: A deep neural network for fake news detection. In *International Conference on Neural Information Processing*, pages 572–581. Springer.

Andreas Vlachos and Sebastian Riedel. 2015. Identification and verification of simple claims about statistical properties. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2596–2601. Association for Computational Linguistics.

Soroush Vosoughi. 2015. *Automatic detection and verification of rumours on Twitter*. Ph.D. thesis, Massachusetts Institute of Technology.

Ke Wu, Song Yang, and Kenny Q Zhu. 2015. False rumors detection on sina weibo by propagation structures. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pages 651–662. IEEE.

Qiao Zhang, Shuiyuan Zhang, Jian Dong, Jinhua Xiong, and Xueqi Cheng. 2015. Automatic detection of rumor on social network. In *Natural Language Processing and Chinese Computing*, pages 113–122. Springer.

Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. 2018. Detection and resolution of rumours in social media: A survey. *ACM Computing Surveys (CSUR)*, 51(2):32.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PloS one*, 11(3):e0150989.

eventAI at SemEval-2019 Task 7: Rumor Detection on Social Media by Exploiting Content, User Credibility and Propagation Information

Quanzhi Li, Qiong Zhang, Luo Si
Alibaba Group, US
Bellevue, WA 98004, USA
{quanzhi.li, qz.zhang, luo.si}@alibaba-inc.com

Abstract

This paper describes our system for SemEval 2019 RumorEval: Determining rumor veracity and support for rumors (SemEval 2019 Task 7). This track has two tasks: Task A is to determine a user's stance towards the source rumor, and Task B is to detect the veracity of the rumor: true, false or unverified. For stance classification, a neural network model with language features is utilized. For rumor verification, our approach exploits information from different dimensions: rumor content, source credibility, user credibility, user stance, event propagation path, etc. We use an ensemble approach in both tasks, which includes neural network models as well as the traditional classification algorithms. Our system is ranked 1st place in the rumor verification task by both the macro F1 measure and the RMSE measure.

1 Introduction

Social media platforms, such as Twitter, Reddit and Facebook, do not always poses authentic information. Rumors sometimes may spread quickly over these platforms (Castillo et al., 2011; Derczynski and Bontcheva, 2014; Qazvinian et al., 2011). A rumor may be defined as a statement whose truth value is unverified or deliberately false (Qazvinian et al., 2011). Rumors usually spread fear or even euphoria, and they may confuse people and cause them to make wrong decisions. Therefore, rumor detection has gained great interest recently. In this paper, we describe the approaches we used in SemEval 2019 RumourEval: Determining rumor veracity and support for rumors (SemEval 2019 Task 7). This task has two subtasks: Task A - user stance classification and Task B - rumor verification.

Stance classification is to determine the attitude of the author of a post message towards a target (Mohammad et al., 2016). In task A, we focus on stance classification of messages towards the truthfulness of rumors in Twitter or Reddit conversations. Each conversation is defined by a source post that initiates the conversation, and a set of replies to it, which form a conversation thread. The goal is to classify each post into one of the four categories: supporting, denying, querying or commenting (SDQC). For this task, we use an ensemble approach, which combines the prediction results from both the traditional learning models, such as SVM, and a neural network model, using the language features extracted from the message text. Task B predicts the veracity of a rumor: true, false, or unverified (i.e., its veracity cannot be verified based on the given information). Each rumor consists of a source post that makes a claim, and a set of replies, directly or indirectly towards the source post. We also employed an ensemble approach on this task, which uses multiple models together to do the veracity prediction.

For more details about these two tasks, please check the task description paper from the task organizers (Gorrell et al., 2019).

2 Related Studies

Rumor detection on social media has been a popular research topic in recent years. The early exploration of this issue started from two special case studies on rumor propagation during natural disasters like earthquakes and hurricanes (Gupta et. al., 2013; Mendoza et al., 2010). Many existing algorithms (Liu et al., 2015; Wu et. al., 2015; Yang et. al., 2012) for debunking rumors followed the work of Castillo et al. (2011). They studied information credibility and proposed a set of features that are able to retrospectively predict if an event is credible.

Stance classification is also an active research area that has been studied in previous work (Lukasik et al., 2016; Zubiaga et al., 2016; Kochkina et al., 2017). A time sequence classification technique has been proposed for detecting the stance against a rumor (Lukasik et al., 2016). Zubiaga et al. (2016) used sequence of label transitions in tree-structured conversations for classifying stance.

Several studies have applied neural networks on the verification of rumors (Ma et al., 2016; Kochkina et al., 2017; Ma et al., 2017); They mainly focus on analyzing the information propagation structure, and have not utilized much information on user credibility. User stance plays an important role in rumor detection. Recent works have employed multi-task learning approaches to jointly learn stance detection and veracity prediction, in order to improve classification accuracy by utilizing the interdependence between them (Ma et al.,; 2018; Kochkina et al., 2018).

3 System Description

We first describe the data set, the word embedding, our message representation method, and then our systems for the two tasks.

Data set quality: Regarding the annotation of the data set, as the task description already pointed out: the overall inter-annotator agreement rate of 63.7% showed the task to be challenging, and easier for source tweets (81.1%) than for replying tweets (62.2%). This means that there are many conflicting or inconsistent labels. This will confuse the learning based models, and make the model and prediction result unstable. When we analyzed the training data set, we found many such examples. To make the labels more consistent, we run an analysis to find the posts that are basically the same or highly similar, but their labels are different. We then mark these posts, and use the same label, the one labeled on the majority of these posts, on them during training. The similarity between two posts is calculated by cosine measure, and the post/message embedding is used in the calculation. The similarity threshold for being considered as similar posts is empirically set as 0.75.

Word embeddings: A popular word embedding data set used by many previous studies is one that is created from Google news articles using word2vec (Mikolov et al., 2013). Because the data in this task are social media messages, mainly tweets, we think a embedding model built specifically from tweets will be more appropriate. Therefore, we used tweets collected from Twitter to train a word embedding model. Only English tweets are used, and about 200 million tweets are used to build the embedding model. Totally, 3 billion words are processed, and word embeddings are generated for 3.5 million unique terms using the word2vec model (Mikolov et al., 2013) and data from (Li et al, 2017). In this task, although some messages are from Reddit, they are similar to tweets, in terms of language style and message structure, since both are social media messages.

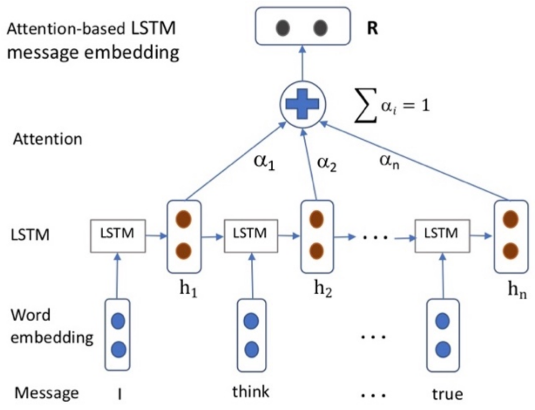


Figure 1. Message embedding based on the attention-based LSTM

Message representation: A tweet (or Reddit message) is usually very shot, consisting of only one sentence. In our models where a whole post is used, such as the stance detection, a message embedding is used as the message representation. We generate the message representation through an attention-based LSTM network for messages that have only one sentence. A post is first preprocessed, such as removing URLs, before it is fed into the LSTM network. Figure 1 shows the network structure for generating the message embedding.

3.1 Stance Detection

Similar to (Kochkina et al., 2017), we use a set of features that include the word embeddings and features generated from the message. They are listed below:

- **Message role:** it is the source message or a reply.
- **Message embedding**: this the message representation presented in Figure 1.
- **Presence of link**: has at least one link or not
- **Link type**: the types of the links. Types include image, video, and article.
- **Relation to source message**: whether this is a reply to the source message
- **Stance of parent message**: if this is a reply message, then the stance of the message it

replies to. In other words, we also check the stance of a message's parent in the propagation path.

- **Similarity with the source message**: Measured by cosine similarity using message embedding.
- **Punctuations**: we check if it has a question mark or exclamation mark.
- **Content length**: the message length after removing links and mentions
- **Mentions:** if the mentions are special accounts, e.g. @cnn, @theonion
- **Hashtags:** if it contains some special hashtags, e.g. #fakenews, #lying

These features are used by the following classification modules: A neural network model. The message embedding and other features are concatenated together and fed into the neural model, which consists of two fully connected layers and a softmax layer for the final label output; Models based on SVM, Random Forest and Logistic Regression, respectively; A rule based model.

The rule-based model handles some special case. Two examples are: 1. For source tweet stance: by default, label the source message as *support*, except A. if it is a Reddit message and it has "debunk this", then label it as *query*. B. if the source message has a question mark, and the sentence has the pattern of asking a question, label it as *query*. For example, the Yes/No questions: "did, do, does, have, has, am, is, are, can, could, may, would, will", and the WH-questions: "what, why, how, when, where", and the as well as their negations. 2. If the message is very short, and mainly contains a couple of keywords or hashtags expressing a very strong opinion, e.g. *#fakenews* or *"not true"*, then a corresponding label is assigned to it.

3.2 Rumor Verification

Our approach for rumor verification utilizes rumor information from several dimensions: text content, user credibility information, rumor propagation path, user stance, etc. Text content is utilized by almost all the previous studies on rumor verification. According to the deception style theory, the content style of deceptive information that aims to deceive readers should be somewhat different from that of the truth, e.g., using exaggerated expressions. An early study from (Castillo et al., 2011) uses many text features in their model. These features and other additional text features are also used in other studies (Liu et al., 2015; Enayet and El-Beltagy, 2017; Ma et al., 2017). Many previous studies have shown that user credibility information is very important in rumor verification (Castillo et al., 2011; Yang et al., 2012; Gupta et al., 2012; Liu et al., 2015; Li et al., 2016; Liu and Wu, 2018). Based on 421 false rumors and 1.47 million related tweets, Li et al. (2016) study various semantic aspects of false rumors, and analyze their spread and user characteristics. Their study shows that user credibility information is a good indicator for judging the credibility of a rumor story. To verify a rumor, we analyze the information from several dimensions:

- **Source content analysis**: whether the source message has links pointing to an article, video or image; length of the source message after removing URLs and mentions; number of named entities, verbs and nouns in the source message; whether the source message contains time expression.
- **Source account credibility:** the following information are considered: is a news agent account; profile has link pointing to top domains; account type: individual or organization (company, government agent, organization); profile has location information; profile has description; profile has image; profile has profession information; is verified user; number of followers, number of posts authors, account age, etc. How they are generated is similar to (Liu et al, 2015).
- **Source account credibility score**: calculated from the information listed in the *Source account credibility* category, and normalized to 0 to 1 (Liu et al, 2015).
- **Reply account credibility:** the profile information to check are similar to the ones in the Source account credibility category.
- **Reply account credibility score:** calculated from the information in the *Reply account credibility category*, and normalized to 0 to 1.
- **Stance of the source message:** get from Task A
- **Stance of replies**: percentage of each stance type; the overall stance score for each stance class, calculated by integrating each reply's account credibility score with its stance.
- **Rumor topic domain**: the topic area of the rumor, e.g. politics, business, science, etc. (Liu et al., 2015; Li et al., 2016).

These data are fed into different models for veracity prediction.

Propagation path analysis: Rumors spread through social media in the form of shares and re-shares of the source post and shared posts, resulting in a diffusion cascade or tree. Each source message has many replies, and they are either direct replies, or replies to other messages in the conversion thread. Take the rumors on Twitter as an example, the training data set contains 327 tweet rumors, and these rumors have 4017 branches and 5570 tweets, which means we have quite a lot replies that are not toward directly to the source message. The structure of the conversion thread is important for understanding the real stance of the user of a reply. For example, given a message "This is fake" and a reply to this message " I totally agree", if we do not know the reply is toward to the first message, then we will give a wrong label "support" to this reply. But actually this reply is denying the rumor claim. This is very important in rumor verification.

Models: We also use the ensemble approach in this task, where multiple models are used to predict rumor veracity, and stacking is employed for the final decision. Similar to the stance detection, we have following classification modules: A neural network model. All the features described above are concatenated together and fed into the neural model, which consists of two fully connected layers and a softmax layer for the final veracity prediction; Three models based on SVM, Random Forest and Logistic Regression, respectively; and a post-processing module that is described below.

Post-processing module

In this component, we consider some special characteristics of rumors and the data sets. We built some simple modules in this components, which may change the prediction results in previous steps. Some of them are described below:

Topics with multiple rumors: Some topics have multiple rumors, and in most cases, the rumors from the same topic have the same veracity, i.e. they are all true, false or unverified. To utilize this characteristic, for a given topic, after each rumor is processed and a prediction is generated, we use this post-process to re-evaluate their veracities, to see if any of them needs to be changed. The rumors from the same topic may or may not talk about the same claim, although in many cases, they are. We calculate the cosine similarity between two source posts, if it is greater than the threshold (set as 0.65 empirically), then the two rumors are considered as basically talking about the same claim, and their veracities are set as the same value. In similarity calculation, a source message is represented by its embedding, already described in previous section. The final veracity is chosen based on the distribution of the veracity values in these rumors, and their corresponding confidence scores.

Rumors originated from special accounts: due to the limited size of training data and annotation quality, some patterns or knowledge may not be caught by the trained models. But when we analyzed the data set, we found some patterns or signals that can provide very strong evidence on rumor veracity, especially for the false rumors. For example, TheOnion is a website usually publishing satirical news and opinions. A source message from this account usually is a false rumor. We check the source post and also the replies in the conversation thread, to see if there is evidence that the claim is from certain special accounts. Another example: if a rumor is from a very credible source, such as news agency NPR or a government agency, then it is very likely its veracity is true.

Rumors debunked or endorsed by special accounts: Similar to the last point, we also check the replies of a source messages, to see if some special accounts have expressed very strong opinion on that claim. For example, if a response is from Snopes.com, a rumor verification website, and it says "#fakenews", or its response is cited by a post in the conversation thread, we can confidently classify this rumor as false. The account information are obtained by analyzing the training data.

Rumor Veracity	Precision	Recall	F measure
True	0.733	0.710	0.721
False	0.828	0.600	0.696
Unverifies	0.227	0.500	0.313
Average	0.596	0.603	0.577

Table 1. Rumer verification result

4 Experiments and Results

The evaluation metric of Task A is the average macro F measure of the four stance categories. Task B uses two evaluation metrics: the average macro F measure of the three veracity types, and also RMSE. Regarding the model training, for the neural network model, the stochastic gradient descent, shuffled mini-batch, AdaDelta update, back-propagation and dropout are used. The word embeddings were fine-tuned during the training process.

For the stance detection task, the result of our system is 0.578 , based on the macro F measure of SDQC. Table 1 shows the rumor detection result of our system. It shows that the unverified category got a very low precision, 0.227, and consequently, its F value is also pretty low, which is 0.313. And this value drags down the average F value of the three categories to 0.577.

References

C. Castillo, M. Mendoza, and B. Poblete. Information credibility on twitter. In Proc. WWW 2011

Ju-han Chuang and Shukai Hsieh. 2015. Stance classification on ptt comments. In Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation.

Genevieve Gorrell, Kalina Bontcheva, Leon Derczynski, Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga, 2019, RumourEval 2019: Determining Rumour Veracity and Support for Rumours. SemEval 2019

Gupta, H. Lamba, P. Kumaraguru, and A. Joshi. Faking sandy: characterizing and identifying fake images on twitter during hurricane sandy. WWW 2013

Elena Kochkina, Maria Liakata, Isabelle Augenstein, 2017, Turing at SemEval-2017 Task 8: Sequential Approach to Rumour Stance Classification with Branch-LSTM, SemEval 2017

Quanzhi Li, Sameena Shah, Xiaomo Liu, Armineh Nourbakhsh, Rui Fang, TweetSift: Tweet Topic Classification Based on Entity Knowledge Base and Topic Enhanced Word Embedding, CIKM 2016

Quanzhi Li, Xiaomo Liu, Rui Fang, Armineh Nourbakhsh, Sameena Shah, User Behaviors in Newsworthy Rumors: A Case Study of Twitter. The 10th International AAAI Conference on Web and Social Media (ICWSM 2016)

Quanzhi Li, Sameena Shah, Xiaomo Liu, Armineh Nourbakhsh, 2017, Data Set: Word Embeddings Learned from Tweets and General Data, The 11th International AAAI Conference on Web and Social Media (ICWSM-2017).

Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, Rui Fang, Sameena Shah, 2015, Real-time Rumor Debunking on Twitter, CIKM 2015.

Yang Liu and Yi-fang Brook Wu. 2018. Early Detection of Fake News on Social Media Through Propagation Path Classification with CNN. AAAI 2018

Michal Lukasik, P. K. Srijith, Duy Vu, Kalina Bontcheva, Arkaitz Zubiaga, and Trevor Cohn. 2016. Hawkes processes for continuous time sequence classification: an application to rumour stance classification in twitter. In Proceedings of the 54th Meeting of the Association for Computational Linguistics. Association for Computer Linguistics, pages 393–398.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks. In Proceedings of IJCAI

Jing Ma, Wei Gao, Kam-Fai Wong, 2017, Detect rumors in microblog posts using propagation structure via kernel learning, ACL 2017

Jing Ma, Wei Gao, Kam-Fai Wong, Detect Rumor and Stance Jointly by Neural Multi-task Learning, WWW 2018

M. Mendoza, B. Poblete, and C. Castillo. Twitter under crisis: Can we trust what we rt? In Proc. First Workshop on Social Media Analytics, 2010.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of NIPS, 2013.

Saif M Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In Proceedings of the International Workshop on Semantic Evaluation, SemEval . volume 16.

V. Qazvinian, E. Rosengren, D. R. Radev, and Q. Mei. Rumor has it: Identifying misinformation in microblogs. EMNLP 2011

Sarvesh Ranade, Rajeev Sangal, and Radhika Mamidi. 2013. Stance classification in online debates by recognizing users' intentions. In Proceedings of the SIGDIAL 2013 Conference . pages 61–69.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. PloS one 11(3):e0150989.

K. Wu, S. Yang, and K. Q. Zhu. False rumors detection on sina weibo by propagation structures. In IEEE International Conference of Data Engineering, 2015.

F. Yang, Y. Liu, X. Yu, and M. Yang. Automatic detection of rumor on sina weibo. In Proc. of the ACM SIGKDD Workshop on Mining Data Semantics, page 13, 2012.

SemEval-2019 Task 8: Fact Checking in Community Question Answering Forums

Tsvetomila Mihaylova,[1] Georgi Karadzhov,[2] Pepa Atanasova,[3]
Ramy Baly,[4] Mitra Mohtarami,[4] Preslav Nakov[5]
[1] Instituto de Telecomunicações, Lisbon, Portugal, [2] SiteGround Hosting EOOD, Bulgaria
[3] University of Copenhagen, Denmark
[4] MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA
[5] Qatar Computing Research Institute, HBKU
{tsvetomila.mihaylova, georgi.m.karadjov}@gmail.com,
pepa@di.ku.dk, {baly,mitram}@mit.edu, pnakov@qf.org.qa

Abstract

We present SemEval-2019 Task 8 on *Fact Checking in Community Question Answering Forums*, which features two subtasks. Subtask A is about deciding whether a question asks for factual information vs. an opinion/advice vs. just socializing. Subtask B asks to predict whether an answer to a factual question is true, false or not a proper answer. We received 17 official submissions for subtask A and 11 official submissions for Subtask B. For subtask A, all systems improved over the majority class baseline. For Subtask B, all systems were below a majority class baseline, but several systems were very close to it. The leaderboard and the data from the competition can be found at http://competitions.codalab.org/competitions/20022.

1 Overview

The current coverage of the political landscape in both the press and in social media has led to an unprecedented situation. Like never before, a statement in an interview, a press release, a blog note, or a tweet can spread almost instantaneously. The speed of proliferation leaves little time for double-checking claims against the facts, which has proven critical in politics, e.g., during the 2016 presidential campaign in the USA, which was dominated by fake news in social media and by false claims.

Investigative journalists and volunteers have been working hard to get to the root of a claim and to present solid evidence in favor or against it. Manual fact-checking is very time-consuming, and thus automatic methods have been proposed to speed-up the process, e.g., there has been work on checking the factuality/credibility of a claim, of a news article, or of an information source (Ba et al., 2016; Zubiaga et al., 2016; Ma et al., 2016; Castillo et al., 2011; Baly et al., 2018).

The process starts when a document is made public. First, an intrinsic analysis is carried out in which check-worthy text fragments are identified. Then, other documents that might support or rebut a claim in the document are retrieved from various sources. Finally, by comparing a claim against the retrieved evidence, a system can determine whether the claim is likely true or likely false (or unsure, if no strong enough evidence either way could be found). For instance, Ciampaglia et al. (2015) do this using a knowledge graph derived from Wikipedia. The outcome could then be presented to a human expert for final judgement.[1]

For our two subtasks, we explore factuality in the context of Community Question Answering (cQA) forums. Forums such as StackOverflow, Yahoo! Answers, and Quora are very popular these days, as they represent effective means for communities around particular topics to share information. However, the information shared by the users is not always correct or accurate. There are multiple factors explaining the presence of incorrect answers in cQA forums, e.g., misunderstanding of the question, ignorance or maliciousness of the responder. Also, as a result of our dynamic world, the truth is time-sensitive: something that was true yesterday may be false today. Moreover, forums are often barely moderated and thus lack systematic quality control.

Here we focus on checking the factuality of questions and answers in cQA forums. This aspect was ignored in recent cQA tasks (Ishikawa et al., 2010; Nakov et al., 2015, 2016a, 2017a), where an answer is considered GOOD if it addresses the question, irrespective of its veracity, accuracy, etc.

[1]As of present, fully automatic methods for fact checking still lag behind in terms of quality, and thus also of credibility in the eyes of the users, compared to what high-quality manual checking by reputable sources can achieve, which means that a final double-checking by a human expert is needed.

Q: HI ;; IF WIFE IS UNDER HER HUSBAND'S SPONSORSHIP AND IS WILLING TO COME QATAR ON VISIT; HOW LONG SHE CAN STAY AFTER EXTENDING THE VISA EVERY MONTH? I HAVE HEARD ITS NOT POSSIBLE TO EXTEND VISIT VISA MORE THAN 6 MONTHS? CAN U PLEASE ANSWER ME.. THANKZZZ...

a_1: Maximum period is 9 Months....

a_2: 6 months maximum

a_3: This has been answered in QL so many times. Please do search for information regarding this. BTW answer is 6 months.

Figure 1: Example from the *Qatar Living* forum.

Figure 1 presents an excerpt of an example from the Qatar Living Forum, with one question and three answers selected from a longer thread. According to SemEval-2016 Task 3 (Nakov et al., 2016a), all three answers would be considered GOOD since they are formally answering the question. Nevertheless, a_1 contains false information, while a_2 and a_3 are correct, as can be established from an official government website.[2]

Checking the veracity of answers in a cQA forum is a hard problem, which requires putting together aspects of language understanding, modelling the context, integrating several information sources, uisng world knowledge and complex inference, among others. Moreover, high-quality automatic fact-checking would offer better experience to users of cQA systems, e.g., the user could be presented with veracity scores, where low scores would warn the user not to completely trust the answer or to double-check it.

2 Related Work

Fact-checking of answers was not studied before in the context of community Question Answering, apart from our own recent work (Mihaylova et al., 2018). Yet, in the context of cQA and general QA, there has been work on *credibility* assessment, which has been modelled primarily at the feature level, with the goal of improving GOOD answer identification. A notable exception are (Nakov et al., 2017b; Mihaylov et al., 2018), where credibility was a task on its own right. However, *credibility* is different from *veracity* (our focus here) as it is a subjective perception about whether a statement is credible, rather than actually truthful.

Jurczyk and Agichtein (2007) modelled author authority using link analysis. Agichtein et al. (2008) looked for high-quality answers using PageRank and HITS, in addition to intrinsic content quality, e.g., punctuation and typos, syntactic and semantic complexity, and grammaticality.

Lita et al. (2005) studied three qualitative dimensions for answers: source credibility (e.g., does the document come from a government website), sentiment analysis, and contradiction compared to other answers. Su et al. (2010) looked for verbs and adjectives that cast doubt. Banerjee and Han (2009) used language modelling to validate the reliability of an answer's source. Jeon et al. (2006) focused on non-textual features such as click counts, answer activity level, and copy counts. Pelleg et al. (2016) curated social media content using syntactic, semantic, and social signals. Unlike this research, we (*i*) target factuality rather than credibility, (*ii*) address it as a task in its own right, and on a specialised dataset.

Information credibility was also studied in social computing. Castillo et al. (2011) modeledd user reputation. Canini et al. (2011) analyzed the interaction of content and social network structure. Morris et al. (2012) studied how Twitter users judge truthfulness. Lukasik et al. (2015) used temporal patterns to detect rumors, and Zubiaga et al. (2016) focused on conversations.

Other authors have been querying the Web to gather support for accepting or refuting a claim (Popat et al., 2016; Karadzhov et al., 2017b). In social media, there has been research targeting the user, e.g., finding malicious users (Mihaylov and Nakov, 2016; Mihaylova et al., 2018; Mihaylov et al., 2018), *sockpuppets* (Maity et al., 2017), *Internet water army* (Chen et al., 2013), and *seminar users* (Darwish et al., 2017).

Finally, there has been work on credibility, trust, and expertise in news communities (Mukherjee and Weikum, 2015). Dong et al. (2015) proposed that a trustworthy source is one that contains very few false claims. Recent work has also focused on evaluating the factuality of reporting of entire news outlets (Baly et al., 2018, 2019).[3] However, none of this work was about QA or cQA.

[2]http://portal.moi.gov.qa/wps/portal/MOIInternet/departmentcommittees/visasentrypermeits/

[3]Knowing the reliability of a medium is important when fact-checking a claim (Popat et al., 2017; Nguyen et al., 2018) and when solving article-level tasks such as "fake news" and click-bait detection (Hardalov et al., 2016; Karadzhov et al., 2017a; Pan et al., 2018; Pérez-Rosas et al., 2018).

3 Subtasks and Data Description

SemEval-2019 Task 8 has two subtasks:

- **Subtask A:** Given a question from a cQA forum, predict whether this question asks for factual information vs. opinion/advice vs. just socializing.

- **Subtask B:** Given a factual question from a cQA forum, together with its answer thread, predict whether each answer provides true vs. false vs. non-factual information as a response to the question.

3.1 Data and Resources

We retrieved the data from the Qatar Living web forum[4]. We then cleaned it and we annotated it with the labels described in Sections 3.2 and 3.3.

For subtask A, we annotated the questions using Amazon Mechanical Turk[5]. To ensure high quality of the annotation, we went through all annotations and manually double-checked them.

For subtask B, we did not use an external annotation service, but instead we annotated all the data ourselves. Each answer was processed by three independent annotators, and we made sure we had proof for the label from reliable sources on the Web. Then, the annotations were consolidated after a discussion until agreement was achieved for each example.

All data is freely available under a Creative Commons Attribution 3.0 Unported (CC BY 3.0) license, and is accessible on the competition's website[6].

In addition to the provided annotated data, we also allowed the participants to use unlabelled data from the Qatar Living forum footnote`http://alt.qcri.org/semeval2016/task3/data/uploads/QL-unannotated-data-subtaskA.xml.zip`, as well as additional external resources, which they had to mention explicitly in their submissions.

Note that the class distribution in the training, development and test sets differs, especially for Subtask B. The reason for this is the way the data was prepared. The different datasets (training, development and test) were prepared on stages, because of the very time-consuming data annotation process.

[4]`http://www.qatarliving.com`
[5]`http://www.mturk.com/`
[6]`http://competitions.codalab.org/competitions/20022`

For each dataset annotation stage, we had to choose between releasing all the available annotated data or aim at releasing sets with similar label distribution. At the end, we decided to release the available data, although we were aware that this would result in releasing sets with different distribution and, in some cases, unbalanced categories.

3.2 Training Data for Subtask A

To create the dataset for the task, we chose to augment a pre-existing dataset for cQA with factuality annotations; this allowed us to stress the difference between (*a*) distinguishing a good vs. a bad answer, and (*b*) distinguishing a factually-true vs. a factually-false one. In particular, we added annotations for factuality to the CQA-QL-2016 dataset from SemEval-2016 Task 3 on Community Question Answering (Nakov et al., 2016a).

In CQA-QL-2016, the data is organized in question–answer threads (from the Qatar Living forum). Each question has a subject, a body, and meta information: question ID, date and time of posting, user name and ID, and category (e.g., *Computers and Internet* and *Moving to Qatar*).

We analyzed the forum questions and we defined three categories, related to their factuality. We then annotated the questions using Amazon Mechanical Turk. The three factuality categories are as follows:

∗ FACTUAL: The question asks for factual information, which can be answered by checking various information sources, and it is not ambiguous (e.g., "*What is Ooredoo customer service number?*").

∗ OPINION: The question asks for an opinion or an advice, not for a fact. (e.g., "*Can anyone recommend a good Vet in Doha?*")

∗ SOCIALIZING: Not a real question, but rather socializing/chatting. This can also mean expressing an opinion or sharing some information, without really asking anything of general interest. (e.g., "*What was your first car?*")

Table 1 shows the distribution of the labels for the question labels in the training, in the development and in the testing datasets. Overall, there are 1118, 239 and 953 questions annotated with the above-described labels.

Label	Train	Dev	Test
FACTUAL	311	62	299
OPINION	563	126	167
SOCIALIZING	244	51	487
TOTAL	**1118**	**239**	**953**

Table 1: **Subtask A:** Distribution of the factuality labels for the questions.

Label	Train	Dev	Test
TRUE	166	29	34
FALSE	135	31	45
NONFACTUAL	194	52	231
TOTAL	**495**	**112**	**310**

Table 2: **Subtask B:** Distribution of the factuality labels for the answers.

3.3 Training Data for Subtask B

For subtask B, we annotated for veracity the answers to the questions with a FACTUAL label for subtask A. Note that in CQA-QL-2016, each answer has a subject, a body, meta information (answer ID, user name, and ID), the question that it answers, and a judgement about how well it answers the question of its thread (GOOD , BAD or POTENTIALLY USEFUL).

We annotated the GOOD answers for factuality based on the assumption that a GOOD answer means it provides factual information, whether it is true or false. All BAD and POTENTIALLY USEFUL answers are automatically considered as NON-FACTUAL. The factuality labels are described as follows:

∗ FACTUAL – TRUE: The answer is True and can be proven with an external resource. (**Q:** *"I wanted to know if there were any specific shots and vaccinations I should get before coming over [to Doha]."*; **A:** *"Yes there are; though it varies depending on which country you come from. In the UK; the doctor has a list of all countries and the vaccinations needed for each."*).[7]

∗ FACTUAL – FALSE: The answer gives a factual response, but it is False and this can be proven using an external resource. (**Q:** *"Can I bring my pitbulls to Qatar?"*; **A:** *"Yes you can bring it but be careful this kind of dog is very dangerous."*).[8]

∗ FACTUAL – PARTIALLY TRUE: The answer contains more than one claim, and only some of these claims could be manually verified. (**Q:** *"I will be relocating from the UK to Qatar [...] is there a league or TT clubs / nights in Doha?"*; **A:** *"Visit Qatar Bowling Center during thursday and friday and you'll find people playing TT there."*).[9]

∗ FACTUAL – CONDITIONALLY TRUE: The answer is True in some cases, and False in others, depending on some conditions that the answer does not mention. (**Q:** *"My wife does not have NOC from Qatar Airways; but we are married now so can i bring her legally on my family visa as her husband?"*; **A:** *"Yes you can."*).[10]

∗ FACTUAL - RESPONDER UNSURE: The person giving the answer is not sure about the veracity of his/her statement. (e.g., "*Possible only if government employed. That's what I heard.*")

∗ NON-FACTUAL: When the answer does not provide factual information to the question; it can be an opinion or an advice that cannot be verified. (e.g., "*Its better to buy a new one.*").

We further discarded answers whose factuality was very time-sensitive and it makes no sense to check whether the statements are true or false (e.g., "*It is Friday tomorrow.*", "*It was raining last week.*").

Moreover, many answers are arguably somewhat time-sensitive, e.g., "*There is an IKEA in Doha.*" is true only after IKEA opened, but not before that. In such cases, we just used the present situation as a point of reference. We further discarded the answers for which the annotators could not find any information.

[7]The answer is factually true and this can be seen at http://wwwnc.cdc.gov/travel/destinations/traveler/none/qatar

[8]The answer is incorrect since pitbulls are included in the list of breeds banned in Qatar. See http://canvethospital.com/pet-relocation/banned-dog-breed-list-qatar-2015/

[9]The place mentioned in the answer has table tennis, but we do not know on which days. See http://www.qatarbowlingfederation.com/bowling-center/

[10]This answer can be true, but this depends upon some conditions. See http://www.onlineqatar.com/info/dependent-family-visa.aspx

Ultimately, we consolidated the above fine-grained labels into the following coarse-grained labels, which we used for subtask B:

∗ FACTUAL – TRUE: Contains answers with proven true, non-contradictory statements. This includes the answers with the label FACTUAL – TRUE from above. This label is used for answers one can trust as a true statement.

∗ FACTUAL – FALSE: Contains answers with statements that are proven to be false or not completely true. This includes answers with the following fine-grained factuality labels: FACTUAL – FALSE, FACTUAL – PARTIALLY FALSE, FACTUAL – CONDITIONALLY TRUE, FACTUAL – RESPONDER UNSURE. We also use this label for answers that contain a statement for which the person giving the answer expresses uncertainty in the claim.

∗ NON-FACTUAL: These are either non-factual statements or statements that could be factual, but no information about them could be found, i.e., we could find no way to check whether the statement was true or false. This category also includes some statements that have been incorrectly annotated as a GOOD answer. It also includes the very time-sensitive statements described before, such as "*It is Friday tomorrow?*". The BAD and the POTENTIALLY USEFUL answers from CQA-QL-2016 also fall in this category.

As we have mentioned above, we have annotated the answers to the FACTUAL questions selected from the Qatar Living forum. We targeted very high quality annotation, and thus we did not use crowd-sourcing, as a pilot experiment has shown that the task was very difficult and that it was not possible to guarantee that Turkers would do all the necessary verification and gather evidence from trusted sources. Instead, all examples were first annotated independently by three of us, and then, we carefully discussed *each example* to come up with a final label. The distribution of the labels on the training, on the development, and on the testind dataset are shown in Table 2[11].

[11]Although not very big, our dataset is larger than datasets used for similar problems, e.g., Ma et al. (2015) experimented with 226 rumors for rumor detection, and Popat et al. (2016) used 100 Wiki hoaxes for credibility assessment of textual claims.

3.4 Evaluation

Both subtasks are three-way classification problems. In subtask A, the questions were to be classified as FACTUAL, OPINION, or SOCIALIZING. Similarly, in subtask B there were also three target categories for the answers: FACTUAL - TRUE, FACTUAL - FALSE, and NON-FACTUAL.

We further scored the submissions based on Accuracy, macro-F1, and average recall (AvgRec).[12] For subtask B, we also report mean average precision (MAP), where the FACTUAL - TRUE instances were considered to be positive, and the remaining ones were negative. The official evaluation measure for both subtasks was Accuracy.

4 Participants and Results

We received 17 official submissions for Subtask A and 11 official submissions for Subtask B. Below we report the evaluation results.

Table 3 presents the results for subtask A on question classification. The results are based the official submissions in the evaluation phase. In this subtask, all of the submitted systems managed to improve over the majority class baseline, and several teams achieved similarly good results. Whenever a number of teams achieve the same result with respect to the main evaluation measure, i.e., Accuracy, we rank them according to the F1 score, and then by AvgRec if a tie still appears.

Table 4 presents the results based on the evaluation phase on the test set for predicting answer factuality labels. This subtask was more difficult as the majority class baseline was very high due to label unbalance. No team managed to improve over that baseline, but several teams had results that were very close to it.

5 Discussion

In the evaluation phase of the competition, the participants had to specify one official submission and were allowed up to two contrastive submissions. In the post-evaluation phase, they could upload an unlimited number of contrastive submissions. Below, we will only discuss the official submissions. The contrastive submissions, the ablation studies, and the experiments with different techniques are described by the participants in their respective system description papers.

[12]Average recall has some attractive properties and has been used in previous SemEval tasks, e.g., (Nakov et al., 2016b; Rosenthal et al., 2017).

Team ID	**Affiliation**	**Accuracy**	**F1**	**AvgRec**
Fermi (Syed et al., 2019)	IIIT Hyderabad, Microsoft, Teradata	**0.840**	0.718_2	0.735_3
TMLab (Niewiński et al., 2019)	Samsung R&D Institute, Warsaw, Poland	**0.834**	0.725_1	0.764_1
SolomonLab (Gupta et al., 2019)	Samsung R&D Institute India, Bangalore	**0.831**	0.709_4	0.728_4
ColumbiaNLP (Chakrabarty and Muresan, 2019)	Columbia University, Department Of Computer Science and Data Science Institute	**0.828**	0.645_7	0.662_9
DOMLIN (Stammbach et al., 2019)	Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Saarbrucken, Germany	**0.823**	0.710_3	0.755_2
BLCU_NLP (Xie et al., 2019)	Beijing Language and Culture University, Beijing, China	**0.820**	0.696_5	0.723_5
pjetro	Warsaw University of Technology	**0.790**	0.661_6	0.698_6
LP0606		**0.768**	0.637_8	0.679_8
PP08		**0.766**	0.637_9	0.684_7
AUTOHOME-ORCA (Lv et al., 2019)	Autohome Inc., Beijing, China and Beijing University of Posts and Telecommunications, Beijing, China	**0.745**	0.583_{10}	0.596_{11}
DUTH (Bairaktaris et al., 2019)	Democritus University of Thrace, Xanthi, Greece	**0.711**	0.563_{11}	0.604_{10}
cococold		**0.702**	0.543_{12}	0.594_{12}
nothing		**0.702**	0.543_{12}	0.594_{12}
chchao		**0.630**	0.454_{13}	0.523_{13}
CodeForTheChange (Avvaru and Pandey, 2019)	International Institute of Information Technology, Hyderabad, Teradata and Qubole	**0.630**	0.442_{14}	0.513_{14}
Tuefact (Juhasz et al., 2019)	University of Tübingen, Tübingen, Germany	**0.599**	0.360_{15}	0.348_{15}
Reem06		**0.549**	0.263_{16}	0.343_{16}
Majority Class Baseline		**0.450**	*0.009*	*0.333*

Table 3: **Subtask A:** Results for question classification based on the official submissions, evaluated on the test set. (Some teams did not submit system description papers, and thus we have no citations for their systems.)

Team ID	**Affiliation**	**Accuracy**	**F1**	**AvgRec**	**MAP**
AUTOHOME-ORCA	Autohome Inc., Beijing, China and Beijing University of Posts and Telecommunications, Beijing, China	**0.815**	0.511_2	0.512_2	0.155_7
ColumbiaNLP	Columbia University, Department Of Computer Science and Data Science Institute	**0.791**	0.524_1	0.635_1	0.134_8
DOMLIN	Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Saarbrucken, Germany	**0.718**	0.402_3	0.445_3	0.267_3
SolomonLab	Samsung R&D Institute India, Bangalore	**0.686**	0.375_4	0.403_4	0.333_2
CodeForTheChange	International Institute of Information Technology, Hyderabad, Teradata and Qubole	**0.654**	0.325_5	0.326_5	0.156_6
BLCU_NLP	Beijing Language and Culture University, Beijing, China	**0.611**	0.296_6	0.317_6	0.222_4
LP0606		**0.548**	0.271_7	0.341_7	0.121_9
PP08		**0.548**	0.271_7	0.341_7	0.121_9
Tuefact	University of Tübingen, Tübingen, Germany	**0.527**	0.260_8	0.347_8	0.571_1
cococold		**0.439**	0.133_9	0.241_9	0.208_5
nothing		**0.439**	0.133_9	0.241_9	0.208_5
Majority Class Baseline		**0.830**	*0.285*	*0.333*	*0.156*

Table 4: **Subtask B:** Results for answer classification based on the official submissions, evaluated on the test set.

The best system for Subtask A was by team Fermi (IIIT Hyderabad). They used Google's Universal Sentence representation (Cer et al., 2018), and XGBoost (Chen and Guestrin, 2016).

The best system for Subtask B was by team AUTOHOME-ORCA (Autohome Inc. and Beijing University of Posts and Telecommunications), who used BERT (Devlin et al., 2019).

They achieved their best results by using an ensemble, and by also using question meta-information (category and subject) in addition to the question and the answer text. They concatenated the category, the subject and the body of the questions into the first part separated by [SEP]. The replier's username and statement were concatenated as the second part. The two parts separated by [SEP] were pushed into the BERT model for answer classification. Then, based on the sequential outputs of the BERT model, some variant methods such as average-pooling, and bi-LSTM were adopted to produce the final results. To tackle the problem with insufficient training data, they further used data augmentation based on translation with Google Translate: in particular, they performed consecutive English-Chinese and Chinese-English translation to generate more synthetic training data.

Overall, the submitted systems for the two subtasks used a number of pre-processing steps to clean the text of the question and of the answer. As shown by the DOMLIN team, the pre-processing of the data turns out to be crucial. They reported up to 5% improvement in terms of accuracy when cleaning the unannotated forum data before fine-tuning a BERT model. Common preprocessing steps included removing or replacing the URLs, the numbers, the punctuation, the symbols, spell-checking, expansion of contractions, HTML tags, etc. DUTH also used lemmatization and stopword removal.

The submitted systems used a wide range of strategies for training their models. A sizable part of the systems used manually crafted features such as linguistic, syntactic, stylistic, and semantic features. Moreover, the systems used task-specific information such as answer ranking and rating. ColumbiaNLP also computed an average cosine similarity of one answer with respect to the other answers in the thread for subtask B, assuming that bad answers would differ substantially from the remaining answers.

While some of the approaches used character and word n-gram information, the teams also used word- and sentence-level embeddings. CodeForTheChange evaluated different classification algorithms fed with Skip-Thought vectors, and ultimately found that neural networks performed best for both subtasks with either concatenation or averaging over the vectors of the available texts. Fermi performed evaluation of different embedding models - InferSent, Concatenated Power Mean Word Embedding, Lexical Vectors, ELMo and The Universal Sentence Encoder, used in subtask A to feed an XGBoost classifier. ColumbiaNLP used ULMFiT, but performed additional unsupervised tuning of the language model on questions, answers and question-answer pairs from the Qatar Living Forum. TMLab's system used the Universal Sentence Encoder.

A common neural network architecture was LSTM, where YNU-HPCC combined LSTM with an attention mechanism. TueFact used comment chain embeddings. Other machine learning algorithms that participants tried include Random Forest, Adaboost, Perceptron, and SVM, inter alia.

While for question classification (subtask A), all the necessary information was contained in the question text and in the metadata, subtask B required additional resources. Most teams used the provided additional unannotated forum data in order to pre-train their language models or to extract more data with weak supervision (DOMLIN). Furthermore, several teams used other means for data augmentation such as SQuAD (BLCU NLP) or external Web information (SolomonLab).

6 Conclusion

We have described SemEval 2019 Task 8 on Fact Checking in Community Question Answering Forums. We received 17 and 11 submissions for Subtask A and B, respectively. Overall, subtask A (question classification) was easier and all submitted systems managed to improve over the majority class baseline. However, Subtask B (answer classification) proved to be much more challenging, and no team managed to improve over the majority class baseline, even though several teams came very close. For this latter subtask, using external resources and preprocessing proved to be crucial.

Acknowledgments

This research is part of the Tanbih project,[13] which aims to limit the effect of "fake news", propaganda and media bias by making users aware of what they are reading. The project is developed in collaboration between the Qatar Computing Research Institute (QCRI), HBKU and the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL).

[13]http://tanbih.qcri.org/

References

Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. 2008. Finding high-quality content in social media. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, pages 183–194, Palo Alto, CA, USA.

Adithya Avvaru and Anupam Pandey. 2019. Code-ForTheChange at SemEval-2019 task 8: Skip-thoughts for fact checking in community question answering. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '19, Minneapolis, MN, USA.

Mouhamadou Lamine Ba, Laure Berti-Equille, Kushal Shah, and Hossam M Hammady. 2016. VERA: A platform for veracity estimation over web data. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 159–162, Montreal, Canada.

Anastasios Bairaktaris, Symeon Symeonidis, and Avi Arampatzis. 2019. DUTH at SemEval-2019 task 8: Part-of-speech features for question classification. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '19, Minneapolis, MN, USA.

Ramy Baly, Georgi Karadzhov, Dimitar Alexandrov, James Glass, and Preslav Nakov. 2018. Predicting factuality of reporting and bias of news media sources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, EMNLP '18, pages 3528–3539, Brussels, Belgium.

Ramy Baly, Georgi Karadzhov, Abdelrhman Saleh, James Glass, and Preslav Nakov. 2019. Multi-task ordinal regression for jointly predicting the trustworthiness and the leading political ideology of news media. In *Proceedings of the 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '19, Minneapolis, MN, USA.

Protima Banerjee and Hyoil Han. 2009. Answer credibility: A language modeling approach to answer validation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL-HLT '09, pages 157–160, Boulder, CO, USA.

Kevin R. Canini, Bongwon Suh, and Peter L. Pirolli. 2011. Finding credible information sources in social networks based on content and social structure. In *Proceedings of the IEEE International Conference on Privacy, Security, Risk, and Trust, and the IEEE International Conference on Social Computing*, SocialCom/PASSAT '11, pages 1–8, Boston, MA, USA.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on Twitter. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 675–684, Hyderabad, India.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Tuhin Chakrabarty and Smaranda Muresan. 2019. ColumbiaNLP at SemEval-2019 task 8: The answer is language model fine-tuning. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '19, Minneapolis, MN, USA.

Cheng Chen, Kui Wu, Venkatesh Srinivasan, and Xudong Zhang. 2013. Battling the Internet Water Army: detection of hidden paid posters. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, pages 116–120, Niagara, Canada.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, San Francisco, California, USA.

Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M. Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. 2015. Computational fact checking from knowledge networks. *PLOS ONE*, 10(6):1–13.

Kareem Darwish, Dimitar Alexandrov, Preslav Nakov, and Yelena Mejova. 2017. Seminar users in the Arabic Twitter sphere. In *Proceedings of the 9th International Conference on Social Informatics*, SocInfo '17, pages 91–108, Oxford, UK.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL-HLT '19, Minneapolis, MN, USA.

Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang. 2015. Knowledge-based trust: Estimating the trustworthiness of web sources. *Proc. VLDB Endow.*, 8(9):938–949.

Ankita Gupta, Sudeep Kumar Sahoo, Divya Prakash, Rohit R R, Vertika Srivastava, and YEON HYANG KIM. 2019. SolomonLab at SemEval-2019 task 8: Question factuality and answer veracity prediction in community forums. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '19, Minneapolis, MN, USA.

Momchil Hardalov, Ivan Koychev, and Preslav Nakov. 2016. In search of credible news. In *Proceedings of the 17th International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, AIMSA '16, pages 172–180, Varna, Bulgaria.

Daisuke Ishikawa, Tetsuya Sakai, and Noriko Kando. 2010. Overview of the NTCIR-8 Community QA Pilot Task (Part I): The Test Collection and the Task. In *Proceedings of NTCIR-8 Workshop Meeting*, pages 421–432, Tokyo, Japan.

Jiwoon Jeon, W. Bruce Croft, Joon Ho Lee, and Soyeon Park. 2006. A framework to predict the quality of answers with non-textual features. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 228–235, Seattle, WA, USA.

Reka Juhasz, Franziska-Barbara Linnenschmidt, and Teslin Roys. 2019. TueFact at SemEval 2019 Task 8: Fact checking in community question answering forums: context matters. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '19, Minneapolis, MN, USA.

Pawel Jurczyk and Eugene Agichtein. 2007. Discovering authorities in question answer communities by using link analysis. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 919–922, Lisbon, Portugal.

Georgi Karadzhov, Pepa Gencheva, Preslav Nakov, and Ivan Koychev. 2017a. We built a fake news & click-bait filter: What happened next will blow your mind! In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, RANLP '17, pages 334–343, Varna, Bulgaria.

Georgi Karadzhov, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, and Ivan Koychev. 2017b. Fully automated fact checking using external sources. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, RANLP '17, pages 344–353, Varna, Bulgaria.

Lucian Vlad Lita, Andrew Hazen Schlaikjer, WeiChang Hong, and Eric Nyberg. 2005. Qualitative dimensions in question answering: Extending the definitional QA task. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20 of *AAAI '05*, pages 1616–1617, Pittsburgh, PA, USA.

Michal Lukasik, Trevor Cohn, and Kalina Bontcheva. 2015. Point process modelling of rumour dynamics in social media. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, ACL-IJCNLP '15, pages 518–523, Beijing, China.

Zhengwei Lv, Duoxing Liu, Haifeng Sun, Xiao Liang, Tao Lei, Zhizhong Shi, Feng Zhu, and Lei Yang. 2019. AUTOHOME-ORCA at SemEval-2019 task 8: Application of BERT for fact-checking in community forums. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '19, Minneapolis, MN, USA.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J. Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI '16, pages 3818–3824, New York, NY, USA.

Jing Ma, Wei Gao, Zhongyu Wei, Yueming Lu, and Kam-Fai Wong. 2015. Detect rumors using time series of social context information on microblogging websites. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1751–1754, Melbourne, Australia.

Suman Kalyan Maity, Aishik Chakraborty, Pawan Goyal, and Animesh Mukherjee. 2017. Detection of sockpuppets in social media. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '17, pages 243–246, Portland, OR, USA.

Todor Mihaylov, Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Georgi Georgiev, and Ivan Koychev. 2018. The dark side of news community forums: Opinion manipulation trolls. *Internet Research*, 28(5):1292–1312.

Todor Mihaylov and Preslav Nakov. 2016. Hunting for troll comments in news community forums. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL '16, pages 399–405, Berlin, Germany.

Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, Mitra Mohtarami, Georgi Karadjov, and James Glass. 2018. Fact checking in community forums. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, AAAI '18, pages 879–886, New Orleans, LA, USA.

Meredith Ringel Morris, Scott Counts, Asta Roseway, Aaron Hoff, and Julia Schwarz. 2012. Tweeting is believing?: Understanding microblog credibility perceptions. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 441–450, Seattle, WA, USA.

Subhabrata Mukherjee and Gerhard Weikum. 2015. Leveraging joint interactions for credibility analysis in news communities. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 353–362, Melbourne, Australia.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017a. SemEval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval '17, pages 27–48, Vancouver, Canada.

Preslav Nakov, Lluís Màrquez, Walid Magdy, Alessandro Moschitti, Jim Glass, and Bilal Randeree. 2015. SemEval-2015 task 3: Answer selection in community question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 269–281, Denver, CO, USA.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016a. SemEval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 525–545, San Diego, CA, USA.

Preslav Nakov, Tsvetomila Mihaylova, Lluís Màrquez, Yashkumar Shiroya, and Ivan Koychev. 2017b. Do not trust the trolls: Predicting credibility in community question answering forums. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, RANLP '17, pages 551–560, Varna, Bulgaria.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016b. SemEval-2016 task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 1–18, San Diego, CA, USA.

An T. Nguyen, Aditya Kharosekar, Matthew Lease, and Byron C. Wallace. 2018. An interpretable joint graphical model for fact-checking from crowds. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, AAAI '18, New Orleans, LA, USA.

Piotr Niewiński, Aleksander Wawer, Maria Pszona, and Maria Janicka. 2019. TMLab SRPOL at SemEval-2019 Task 8: Fact checking in community question answering forums. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '19, Minneapolis, MN, USA.

Jeff Z. Pan, Siyana Pavlova, Chenxi Li, Ningxi Li, Yangmei Li, and Jinshuo Liu. 2018. Content based fake news detection using knowledge graphs. In *Proceedings of the International Semantic Web Conference*, ISWC '18, pages 669–683, Monterey, CA, USA.

Dan Pelleg, Oleg Rokhlenko, Idan Szpektor, Eugene Agichtein, and Ido Guy. 2016. When the crowd is not enough: Improving user experience with social media through automatic quality analysis. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, CSCW '16, pages 1080–1090, San Francisco, CA, USA.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic detection of fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*, COLING '18, pages 3391–3401, Santa Fe, NM, USA.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2016. Credibility assessment of textual claims on the web. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 2173–2178, Indianapolis, IN, USA.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2017. Where the truth lies: Explaining the credibility of emerging claims on the Web and social media. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17, pages 1003–1012, Perth, Australia.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th Internationafon*, SemEval '17, pages 502–518, Vancouver, Canada.

Dominik Stammbach, Stalin Varanasi, and Guenter Neumann. 2019. DOMLIN at SemEval-2019 Task 8: Automated fact checking exploiting user ratings in community question answering forums. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '19, Minneapolis, MN, USA.

Qi Su, Helen Kai yun Chen, and Chu-Ren Huang. 2010. Incorporate credibility into context for the best social media answers. In *Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation*, PACLIC '10, pages 535–541, Sendai, Japan.

Bakhtiyar Syed, Vijayasaradhi Indurthi, Manish Shrivastava, Manish Gupta, and Vasudeva Varma. 2019. Fermi at SemEval-2019 task 8: An elementary but effective approach to question discernment in community qa forums. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '19, Minneapolis, MN, USA.

Wanying Xie, Mengxi Que, Ruoyao Yang, Chunhua Liu, and Dong Yu. 2019. BLCU_NLP at SemEval-2019 task 8: A contextual knowledge-enhanced GPT model for fact checking. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '19, Minneapolis, MN, USA.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PloS one*, 11(3):e0150989.

AUTOHOME-ORCA at SemEval-2019 Task 8: Application of BERT for Fact-Checking in Community Forums

Zhengwei Lv[1] **Duoxing Liu**[1] **Haifeng Sun**[2] **Xiao Liang**[1]

Tao Lei[1] **Zhizhong Shi**[1] **Feng Zhu**[1] **Lei Yang**[1]

[1] Autohome Inc., Beijing, China
[2] Beijing University of Posts and Telecommunications, Beijing, China

{lvzhengwei,liuduoxing,liangxiao12030,leitao,shizhizhong,zhufeng,yanglei}@autohome.com.cn
hfsun@bupt.edu.cn

Abstract

Fact checking is an important task for maintaining high quality posts and improving user experience in Community Question Answering forums. Therefore, the SemEval-2019 task 8 is aimed to identify factual question (subtask A) and detect true factual information from corresponding answers (subtask B). In order to address this task, we propose a system based on the BERT model with meta information of questions. For the subtask A, the outputs of fine-tuned BERT classification model are combined with the feature of length of questions to boost the performance. For the subtask B, the predictions of several variants of BERT model encoding the meta information are combined to create an ensemble model. Our system achieved competitive results with an accuracy of 0.82 in the subtask A and 0.83 in the subtask B. The experimental results validate the effectiveness of our system.

1 Introduction

The Community Question Answering (CQA) forums are gaining more and more popularity because they can offer great opportunity for users to get appropriate answers to their questions from other users. Meanwhile, the accumulated massive questions and answers in CQA forums present a new challenge to provide valuable information for users more effectively. Therefore, researchers have shown an increased interest in CQA systems (Srba and Bielikova, 2016; Wang et al., 2018), aiming to facilitate efficient knowledge acquisition and circulation. Specifically, a large portion of researches mainly focus on the two tasks: find relevant questions to a new question to reuse corresponding answers (*Question Retrieval*), and search for relevant answers among existing answers to other questions (*Answer Selection*).

Despite a great deal of research on CQA, there are relatively few studies focusing on the quality of questions and answers. Actually, the credibility of answers is an important aspect, which can directly affect the user experience for CQA forums. In order to check the veracity of answers automatically, some recent works (Karadzhov et al., 2017; Mihaylova et al., 2018) attempt to utilize external sources and extract appropriate features for classification. Considering the importance of information veracity in CQA forums, the fact checking of answers is still an issue that is worth investigating further.

Therefore, the SemEval-2019 task 8 aims to conduct fact checking in CQA forums. In order to detect the veracity of answers, it is necessary to identify whether the questions are factual firstly. The task is comprised of two subtasks: the subtask A is targeted to identify whether a question is asking for factual information, an opinion/advice or socializing. Given factual questions, the subtask B is aimed to determine whether the corresponding answers are true, false or not factual.

In order to address the SemEval-2019 task 8, we propose a system based on the BERT model (Devlin et al., 2018). In our system, we extend BERT for integrating some meta information of questions into the BERT encoder, and generate an ensemble model from some potential classification models to achieve very competitive results. To be specific, in subtask A, two outputs of fine-tuned BERT classifiers are obtained from subjects and bodies of questions respectively. Then by combining both outputs with the length of questions as features, the AdaBoost method (Schapire, 1999) is utilized to boost the performance of question classification. As for subtask B, while encoding additional meta information (category and subject of questions) into BERT model, we adopt the bagging method for some variants of BERT model produced by adding additional layers. The experimental results in both subtasks demonstrate the effectiveness of our system.

The rest of our paper is organized in the following way. The related work about CQA is summarized in Section 2. Section 3 gives a more detailed description of our system. The results and analysis of experiments are demonstrated in Section 4. Finally, Section 5 presents the main conclusions.

2 Related Work

So far, most studies about CQA mainly pay attention to two tasks: *Question Retrieval* and *Answer Selection*. In previous works, some traditional methods treat questions or answers as bag of words and measure their similarities based on weighted matching between the words (Robertson et al., 1994) or translation probability learning from language model (Xue et al., 2008). In fact, similar questions often are not phrased with exactly same words, but related words, while there is very little token overlap between questions and answers. These methods essentially consider the question or answer as a bag of words, neglecting semantic information. So it is not surprising that the performance of traditional methods is not very well on aforementioned tasks. Recently, the neural-based models (He et al., 2015; Feng et al., 2015; Tan et al., 2016; Bachrach et al., 2017; Tay et al., 2018), which can capture some semantic relations, are proposed and become mainstream in the research about CQA gradually. The basic idea behind them is to learn the representation of questions and answers based on CNN or LSTM models, then conduct text matching by regarding both tasks as classification or learning to rank.

Furthermore, there are also public CQA datasets and competitions available, which promote relevant researches substantially. The public datasets are collected from various CQA websites, including Quora[1] , Yahoo! Answers[2], Qatar Living[3], etc. As for competitions, there is a kaggle competition[4] to identify the duplicated question pairs collecting from the Quora website. In SemEval-2015 Task 3 ”Answer Selection in Community Question Answering” (Nakov et al., 2015), it is mainly targeted on the answer selection task. And there is a more comprehensive competition in SemEval-2016 Task 3 (Nakov et al., 2016) designed for both Question Retrieval and Answer Selection, which is consisted of four subtasks: Question-Comment Similarity, Question-Question Similarity, Question-External Comment Similarity and Reranking the correct answers for a new question. In contrast, in SemEval-2017 task 3 (Nakov et al., 2017), a new duplicate question detection subtask is incorporated on the basis of the SemEval-2016 Task 3.

Although much work has been done in CQA researches, few attentions have been paid on improving the quality of questions and answers. In order to detect true factual answers automatically, Karadzhov et al. (Karadzhov et al., 2017) propose a general framework using external sources, which adopts the LSTM model (Hochreiter and Schmidhuber, 1997) to learn text representation of answers and external sources. Mihaylova et al. (Mihaylova et al., 2018) extract features from multiple aspects (the answer content, the author profile, the rest of the community forum and external authoritative sources) and demonstrate the effectiveness of fact checking of answers. At the same time, the lack of large-scale dataset also restricts the progress on fact checking in CQA forums further.

Recently, there are some of key milestones in the NLP field, such as ELMo (Peters et al., 2018), ULMFiT (Howard and Ruder, 2018), OpenAI GPT (Radford, 2018) and BERT (Devlin et al., 2018). These large-scale models have provided great performance on various NLP tasks, which can be pre-trained on a massive corpus of unlabeled data, and then fine-tuned to downstream tasks. Especially, the BERT model has achieved state-of-the-art results on a variety of language tasks, which allows us to obtain significantly higher performance than models that are only able to leverage a small task-specific dataset. Therefore, we build a system based on the BERT model for the SemEval2019 task 8 and achieve satisfactory results.

3 System Description

3.1 System Overview

The pipeline of our system is shown in Figure 1. Firstly, original input files with questions and answers are preprocessed, including removing redundant information (e.g., HTML tags, URLs and strings exceeding maximum length limit) and extracting the structured contents and

[1]https://www.quora.com
[2]https://answers.yahoo.com
[3]https://www.qatarliving.com
[4]https://www.kaggle.com/c/quora-question-pairs

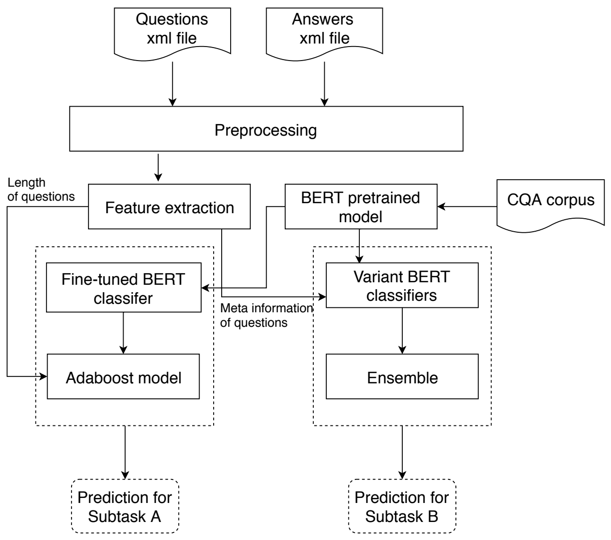


Figure 1: Pipeline of our system.

Label	Subject of questions	Body of questions
Opinion	e.g., *does anyone know good dentist?*	e.g., *can anybody recommend me a dentist? a good one.*
Factual	e.g., *when is eid gonna start?*	e.g., *when will eid start? like holidays*
Socializing	e.g., *What do you like about the person above you?*	e.g., *Hello people...let's play this game...you have to write something good about the person whose 'post' is above you on QL.You can write anything and you can write multiple times. For ex;the person who will respond to my post will write about me ;) and so on. This will be fun...*

Table 1: Samples of questions with different labels.

meta information. Secondly, some important features are obtained from structured information, such as the length and category of questions. Thirdly, based on the pre-trained BERT model released by Google[5], we conduct unsupervised training on specific CQA corpus further to make the model more suitable for the following classification tasks. Finally, the pre-trained BERT model and extracted features are fed into two subsystems to obtain predictions for the subtask A and the subtask B respectively. In the subsystem for subtask A (detailed in Subsection 3.2), the AdaBoost model is adopted to predict the classification of questions by combining the outputs of fine-tuned BERT classifier and the feature of length of questions. In the subsystem for subtask B (described in Subsection 3.3), some variant BERT models which encode meta information of questions are combined to generate an ensemble model for predication of labels of answers.

3.2 Subsystem for Subtask A

In this Subsection, the subsystem for subtask A is described in detail below.

Firstly, the subject and body of questions are encoded into two BERT models separately for fine-tuning on the question classification. The different inputs for both BERT models are represented as

$$[CLS] + text1 + [SEP]$$
$$[CLS] + text2 + [SEP]$$

where $text1$ and $text2$ are the subject and body of question respectively.

Secondly, the outputs of two fine-tuned BERT models are concatenated with the length of questions' body as features for classification. As illustrated in Table 1, it is rather intuitive that the body length of questions for socializing is inclined to be longer than ones for factual or opinion. Therefore, it is reasonable to consider the body length of questions as a suitable feature for classification. In addition, the results of each BERT model are probabilities of questions belonging to different classes (Factual, Opinion and Socializing). Then the feature vector x_{vector} for question classification is represented as follows

$$x_{vector} = [P_s1, P_s2, P_s3, P_b1, P_b2, P_b3, L_b] \quad (1)$$

where P_s1, P_s2, P_s3 are the output of a BERT model encoding the question subject. Similarly, P_b1, P_b2, P_b3 are the output of another BERT model encoding the question body, and L_b is the body length of a question.

Finally, based on the generated feature vector x_{vector}, the AdaBoost algorithm is adopted to obtain the final results of classification. AdaBoost is a typical Boosting algorithm that aims to convert a set of relative weak classifiers to a strong classifier. Therefore, the performance of classification can be strengthened by considering additional length feature, compared to the one that the BERT models

[5]https://github.com/google-research/bert

have achieved.

3.3 Subsystem for Subtask B

The Subsection describes the details of the subsystem for subtask B as follows.

Firstly, the subject and body of question, corresponding reply (i.e., answer) and meta information of question are combined to generate sequences for BERT encoders. In order to identify the true factual reply, the content of corresponding questions and auxiliary information (e.g., the category of question, username of a questioner or replier) should be necessary for classifiers. So in the subsystem, we investigate the influence of different information for the classification performance (see Table 4 for details), including the subject of question (F-subject), the usernames of questioner and replier (F-username) and the category of question (F-category). Ultimately, the text of answer and the information of F-subject and F-category are employed for our BERT based models. The generated sequence for inputs of models are represented as following:

$$[CLS] + text1 + [SEP] + text2 + [SEP]$$

We use [SEP] to separate between the information of question and answer. $text1$ is composed of F-subject, F-category and the body of question separated by the special symbol ($\sim$), while $text2$ is the text of corresponding reply.

Secondly, based on the generated sequences as inputs, we design three different categories of BERT based models for ensemble. As shown in

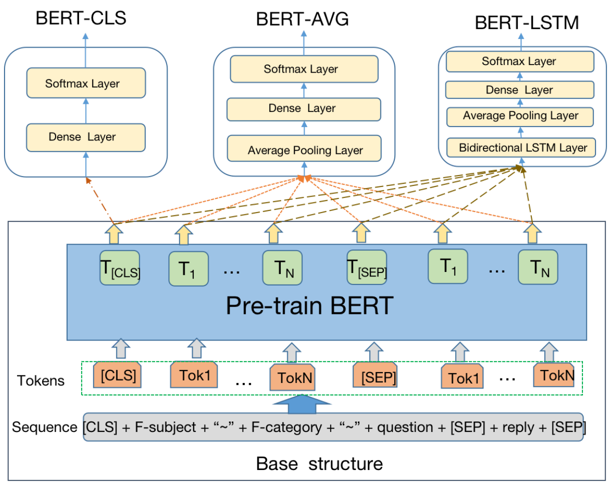


Figure 2: Architecture of BERT based models.

Figure 2, the specific structures of the three kinds of models are described as follows:

- *BERT-CLS*. The final hidden state for the first token [CLS] in the input is employed for fine-tuning the pre-trained BERT model by adding a classification layer and a standard softmax.

- *BERT-AVG*. Different from BERT-CLS, the final hidden states for all tokens are utilized for classification by conducting an average pooling, and then adding a full connected layer and a standard softmax.

- *BERT-LSTM*. Compared with BERT-AVG, a Bi-LSTM network is added between the pooling layer and the pre-trained BERT encoder. It must be noted that we only obtain the outputs of BERT encoder and the parameters of BERT encoder are not updated when training.

Thirdly, we select a set of competitive classifiers in the training process by the three kinds of BERT based models respectively. The method of five-fold cross-validation is employed. To be specific, the original samples are randomly divided into five sub-samples with equal size. And one of the five sub-samples is retained for validating the performance of classifier, and the rest of four sub-samples are used as training data. For each kind of BERT based model, the cross-validation process is repeated five times and each time no more than five optimal classifiers are obtained. Therefore, we get a total of sixty-five competitive classifiers filtered by certain threshold value on accuracy metric from three kinds of BERT based models for ensemble.

Finally, an effective integration strategy is applied to produce a strong classifier for the subtask. There are two candidate integration strategies:

- *Strategy 1 (Vote-ensemble)*: Each classifier casts a vote, the label of a sample is decided according to the majority of votes.

- *Strategy 2 (Distribution-ensemble)*: If the number of votes for any label exceeds one-half of the total number of classifiers, the sample is classified as the corresponding label. Otherwise, the label of the sample will be determined by considering the actual label distribution of the training data and the label distribution of votes together. For example, if one sample's votes for different labels are very close, then the sample is classified as the label with the largest proportion of data distribution.

At last, the strategy 2 is employed in our subsystem because it seems that Distribution-ensemble strategy is more robust for variance error, especially for small dataset, which will be discussed in Subsection 4.2 further.

4 Experiment

4.1 Dataset

The dataset is organized in question-answer threads from the Qatar Living forum. Each question, which is annotated by labels: Opinion, Factual and Socializing, has a subject, a body and meta information including question ID, category, posting time, user's ID and name. And each answer, which is classified as Factual-True, Factual-False and Non-Factual, has a body and meta information (answer ID, posting time, user's ID and name). The detailed statistics of the dataset in this task are illustrated in the task description paper (Mihaylova et al., 2019).

4.2 Experimental Results and Analysis

As for pre-training the BERT model, it is trained based on the BERT-Base-Cased model by the forum corpus provided by organizer[6]. The training batch size is 32, the number of train steps is 1e+5 and the learning rate is 2e-5. The detailed experimental results for both subtasks are described as following.

4.2.1 Results for Subtask A

In the subsystem for subtask A, the AdaBoost algorithm is employed to boost the performance on question classification. The number of estimators for the AdaBoost method is 10. To evaluate the performance of question classification, we compare our proposed method against the following models:

- Text-CNN (Kim, 2014): a simple CNN with one layer of convolution on top of word vectors. The subject and body of each question are concatenated as the input of Text-CNN model. When training, the number of epoch is 80, the initial learning rate is 0.001 and the dropout rate is set to 0.4.

- BERT without pre-training: the BERT-Base cased model release by Google. The input of the model is the concatenation of the subject and the body of each question with the symbol [SEP], which is represented as follows:

$$[CLS] + text1 + [SEP] + text2 + [SEP]$$

  $text1$ and $text2$ are the subject and body of a question separately. When training the model, the batch size of training is 32, the initial learning rate is 2e-5 and the number of epoch is 9.

- BERT with pre-training: the BERT model pre-trained by CQA corpus. The settings of hyper-parameters is the same as the BERT model without pre-training.

Models	Acc. (Dev)	Acc. (Test)
Text-CNN	0.6569	0.6502
BERT without pre-training	0.6862	0.7370
BERT with pre-training	0.7197	0.7922
Our method	**0.7283**	**0.8181**

Table 2: Performance of different models in the subtask A.

The comparison results are shown in Table 2. From the table, it can be observed that the accuracy of the Text-CNN model is much lower than the other three BERT-based models. Even if only the BERT model without pre-training is used to predict the final result, it is 2.93% and 8.68% higher than Text-CNN model on development dataset and test dataset, respectively. Considering the size of dataset is relative small, it seems to demonstrate the potential advantage of BERT based models. Compared with the BERT model without pre-training, the BERT model with pre-training has 3.35% and 5.52% increase respectively. It is illustrated that the step of pre-training the BERT model is very important. Furthermore, the accuracy achieved by our method is 0.86% and 2.59% higher than the one by the BERT with pre-training model on two datasets separately. It shows that the AdaBoost algorithm can make better use of the probability outputs from the fine-tuned BERT models for prediction. What's more, the body length of questions can be considered as an effective feature for training model and predicting results.

[6]http://alt.qcri.org/semeval2016/task3/data/uploads/QL-unannotated-data-subtaskA.xml.zip

4.2.2 Results for Subtask B

In the experiments for subtask B, the three kinds of BERT based models are implemented with TensorFlow and trained with Adam optimizer. The maximum length of sequence is set to 150 and the batch size is 4. The initial learning rates are 3e-5 for parameters of BERT encoder and 1e-3 for others.

Models	Acc.(Dev)	Acc.(Test)
BERT-AVG	0.6732	–
BERT-CLS	0.6667	–
BERT-LSTM	0.656	–
Vote-ensemble	0.6693	0.7935
Distr.-ensemble	**0.6845**	**0.8322**

Table 3: Performance of different models in the subtask B.

The experimental results of different kinds of models are shown in Table 3. From the table, it can be observed that the BERT-AVG model achieves the best performance in the three single models. By conducting average pooling operation on final hidden states of all tokens, the BERT-AVG model can capture more semantic information than the BERT-CLS model which can only pay attention to the hidden state of the [CLS] token. As for the BERT-LSTM model, it performs the worst, which may be caused by the highest model complexity and the lack of adequate training dataset, resulting in somewhat overfitting. In addition, it is indicated that ensemble models can obtain higher accuracy than single models and the strategy of Distribution-ensemble is more robust than the strategy of Vote-ensemble. This is because that when the numbers of votes for different labels are close to each other, it is difficult to identify the correct class only by the majority. By considering actual classification distribution in training dataset additionally, the Distribution-ensemble can show its potential advantage.

Feature	Acc.(Dev)
Baseline	0.6559
+F-category	0.6606(+0.47)
+F-username	0.6547(-0.12)
+F-subject	0.6642(+0.83)
+F-category, +F-subject	**0.6667(+1.08)**

Table 4: Performance of different features on development dataset in the subtask B. "+" means to add current features to the main feature.

In order to explore the effectiveness of different information for classification, a series of experiments based on the BERT-CLS model are conducted. The baseline model (BERT-CLS) is established only by encoding the information of the body of question and the corresponding answer. Therefore, the influence of other information can be discussed individually. By considering different information, the performance of the model validated on development dataset is shown in Table 4. It is observed that the F-username can not contribute to the increase of accuracy, which may be caused by existing many anonymous users in the forum. By encoding the information F-subject and F-category into the model, it can achieve the best performance.

5 Conclusion

Detecting the veracity of answers is vital to maintain high quality information in CQA forums. In order to address this problem, a system based on BERT model is developed for participating in the SemEval-2019 Task 8. In the system, the meta information of questions is encoded into the BERT model and an ensemble with multiple variants of BERT model are produced to accomplish better performance. In subtask A, we utilize the AdaBoost algorithm to the features that is consisted of fine-tuned results of BERT models and length of questions. In subtask B, after encoding the auxiliary information of questions and answers into the BERT model, fine-tuned BERT model and two variant models by adding average-pooling or LSTM layers are combined to reduce the variance error. Finally, our system achieved great performance with an accuracy of 0.82 and 0.83 in the two subtasks respectively.

To our surprise, the system has impressive results in the subtask B without using external sources. It may be explained by the potential advantage of BERT model over other models only trained on a small task-specific dataset. In the future, we will explore to retrieve relevant information from the Web efficiently and then integrate the external information into our BERT based model.

References

Yoram Bachrach, Andrej Zukov Gregoric, Sam Coope, Ed Tovell, Bogdan Maksak, José Rodríguez, Conan McMurtie, and Mahyar Bordbar. 2017. An attention mechanism for neural answer selection using a combined global and local view. *2017 IEEE 29th*

International Conference on Tools with Artificial Intelligence (ICTAI), pages 425–432.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 813–820.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339. Association for Computational Linguistics.

Georgi Karadzhov, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, and Ivan Koychev. 2017. Fully Automated Fact Checking Using External Sources. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 344–353. INCOMA Ltd.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Tsvetomila Mihaylova, Georgi Karadzhov, Atanasova Pepa, Ramy Baly, Mitra Mohtarami, and Preslav Nakov. 2019. SemEval-2019 task 8: Fact checking in community question answering forums. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '19, Minneapolis, MN, USA.

Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, Mitra Mohtarami, Georgi Karadzhov, and James Glass. 2018. Fact Checking in Community Forums. In *AAAI Conference on Artificial Intelligence*.

Preslav Nakov, Lluís Arquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak Abed, Alhakim Freihat, James Glass, Bilal Randeree, and Qatar Living. 2016. SemEval-2016 Task 3: Community Question Answering. In *Proceedings of SemEval-2016*, pages 525–545.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. SemEval-2017 Task 3: Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 27–48.

Preslav Nakov, Lluís Màrquez, Walid Magdy, Alessandro Moschitti, Jim Glass, and Bilal Randeree. 2015. Semeval-2015 task 3: Answer selection in community question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 269–281. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*, pages 109–126.

Robert E. Schapire. 1999. A brief introduction to boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, pages 1401–1406, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Ivan Srba and Maria Bielikova. 2016. A Comprehensive Survey and Classification of Approaches for Community Question Answering. *ACM Transactions on the Web*, 10(3):1–63.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 464–473. Association for Computational Linguistics.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-cast attention networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &#38; Data Mining*, KDD '18, pages 2299–2308, New York, NY, USA. ACM.

Xianzhi Wang, Chaoran Huang, Lina Yao, Boualem Benatallah, and Manqing Dong. 2018. A survey on expert recommendation in community question answering. *Journal of Computer Science and Technology*, 33(4):625–653.

Xiaobing Xue, Jiwoon Jeon, and W. Bruce Croft. 2008. Retrieval models for question and answer archives. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 475–482, New York, NY, USA. ACM.

SemEval-2019 Task 9: Suggestion Mining from Online Reviews and Forums

Sapna Negi
Genesys Telecommunication Laboratories
Galway, Ireland
sapna.negi1@gmail.com

Tobias Daudert
National University of Ireland
Galway, Ireland
tobias.daudert@insight-centre.org

Paul Buitelaar
National University of Ireland
Galway, Ireland
paul.buitelaar@insight-centre.org

Abstract

We present the pilot SemEval task on Suggestion Mining. The task consists of subtasks A and B, where we created labeled data from feedback forum and hotel reviews respectively. Subtask A provides training and test data from the same domain, while Subtask B evaluates the system on a test dataset from a different domain than the available training data. 33 teams participated in the shared task, with a total of 50 members. We summarize the problem definition, benchmark dataset preparation, and methods used by the participating teams, providing details of the methods used by the top ranked systems. The dataset is made freely available to help advance the research in suggestion mining, and reproduce the systems submitted under this task.

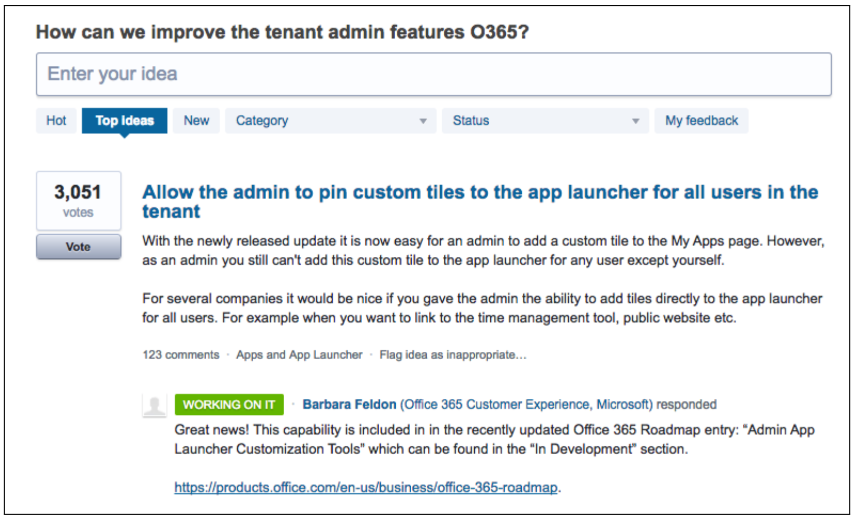
1 Introduction

State of the art opinion mining systems provide numerical summaries of sentiments and tend to overlook additional descriptive and potentially useful content present in the opinionated text. We stress that such content also encompass information like suggestions, tips, and advice, which is otherwise explicitly sought by the stakeholders. For example, hotel reviews often contain room tips, i.e., which room should be preferred in a hotel. Likewise, tips on restaurants, shops, sightseeing, etc. are also present within the hotel reviews. On the other hand, platforms like Tripadvisor[1], which collect hotel and restaurant related opinions, request the reviewers to fill up the room tips section in addition to the hotel review. Likewise, sentences expressing advice, tips, and recommendations relating to a target entity can often be present in text available from different types of data sources, like blogs, microblogs, discussions, etc. Such sentences can be collectively referred to as suggestions. With the increasing availability of opinionated text, methods for automatic detection of suggestions can be employed for different use cases. Some example use cases are the extraction of suggestions for brand improvement, the extraction of tips and advice for customers, the extraction of the expressions of recommendations from unstructured data in order to aid recommender systems, or the summarisation of suggestion forums where suggestion providers often tend to provide context in their responses (Figure 1) which gets repetitive over a large number of responses relating to the same entity. The task of automatic identification of suggestions in a given text is referred to as *suggestion mining* (Brun and Hagege, 2013).

Studies performed on suggestion mining have defined it as a sentence classification task, where class prediction has to be made on each sentence of a given text, classes being *suggestion* and *non suggestion* (Negi, 2016). State of the art opinion mining systems have mostly focused on identifying sentiment polarity of the text. Therefore, suggestion mining remains a very less explored problem as compared to sentiment analysis, specially in the context of recent advancements in neural network based approaches for feature learning and transfer learning.

As suggestion mining is still an emerging research area, it lacks benchmark datasets and well defined annotation guidelines. A few early works were mostly rule based methods, mainly targeted towards the use case of extracting suggestions for product improvements (Brun and Hagege, 2013; Ramanand et al., 2010; Moghaddam, 2015). In our prior work, we performed early investigations on the problem definition and datasets, aiming for the statistical methods which also require benchmark train datasets in addition to the evaluation

[1] https://www.tripadvisor.com



How can we improve the tenant admin features O365?

3,051 votes

Allow the admin to pin custom tiles to the app launcher for all users in the tenant

With the newly released update it is now easy for an admin to add a custom tile to the My Apps page. However, as an admin you still can't add this custom tile to the app launcher for any user except yourself.

For several companies it would be nice if you gave the admin the ability to add tiles directly to the app launcher for all users. For example when you want to link to the time management tool, public website etc.

123 comments · Apps and App Launcher · Flag idea as inappropriate...

Barbara Feldon (Office 365 Customer Experience, Microsoft) responded

Great news! This capability is included in in the recently updated Office 365 Roadmap entry: "Admin App Launcher Customization Tools" which can be found in the "In Development" section.

https://products.office.com/en-us/business/office-365-roadmap.

Figure 1: A post from the suggestion forum for Microsoft developers

datasets(Negi and Buitelaar, 2015; Negi et al., 2016). A few other works also evaluated statistical classifiers (Wicaksono and Myaeng, 2012; Dong et al., 2013), which employed mostly manually identified features, however only two other works (Wicaksono and Myaeng, 2012; Dong et al., 2013) provided their datasets. Suggestion mining still lacks well defined annotation guidelines, a multi-domain and cross-domain approach to the problem and benchmark datasets, which we address in our recent work (Negi et al., 2018). Therefore, we introduce this pilot shared task to disseminate suggestion mining benchmarks and evaluate state of the art methods for text classification on domain specific and cross domain training scenarios. The datasets released as a part of the shared task include the domains hotel reviews and software developers suggestion forum (see Table 1).

Suggestion mining faces similar text processing challenges as other sentence or short text classification tasks related to opinion mining and subjectivity analysis, such as stance detection (Mohammad et al., 2016), or tweet sentiment classification (Rosenthal et al., 2015). Some of the observed challenges in suggestion mining are elaborated below:

- **Class imbalance:** Usually, suggestions tend to appear sparsely among opinionated text, which leads to higher data annotation costs and results in a class distribution bias in the trained models.

- **Figurative expressions:** Text from social media and other sources usually contains figurative use of language, which demands pragmatic understanding from the models. For example, 'Try asking for extra juice at breakfast - its 22 euros!!!!!' is more of a sarcasm than a suggestion. Therefore, a sentence framed as a typical suggestions may not always be a suggestion and vice versa. A variety of linguistic strategies used in suggestions also make this task interesting from a computational linguistics perspective and labeled datasets can be leveraged for linguistic studies as well.

- **Context dependency:** In some cases, context plays a major role in determining whether a sentence is a suggestion or not. For example, 'There is a parking garage on the corner of the Forbes showroom.' can be labeled as a suggestion (for parking space) when it appears in a restaurant review and a human annotator gets to read the full review. However, the same sentence would not be labeled as a suggestion if the text is aimed to describe the surroundings of the Forbes showroom.

- **Long and complex sentences:** Often, a suggestion is expressed in either one part of a sentence, or it is elaborated as a long sentence, like, *'I think that there should be a nice feature where you can be able to slide the status bar down and view all the push notifications that you got but you didn't view, just like*

Source	Sentence	Label
Hotel reviews	Be sure to specify a room at the back of the hotel.	suggestion
Hotel reviews	The point is, don't advertise the service if there are caveats that go with it.	non-suggestion
Suggestion forum	Why not let us have several pages that we can put tiles on and name whatever we want to	suggestion
Suggestion forum	It fails with a uninformative message indicating deployment failed.	non-suggestion

Table 1: Examples of suggestions found in reviews and the labels assigned to the suggestion sentences

android and IOS, but the best part is that it fixes many problems like when people wanted a short cut to turn WiFi on and off and data on and off so that would be a nice feature to have 2'. This poses challenges to the training of algorithms in terms of learning effective features, as well as for certain pre-processing steps like part of speech tagging.

Investigating the development of high performance suggestion mining systems could drive the engagement of both, commercial entities (like brand owners) as well as the research communities, working on problems such as opinion mining, supervised learning, or representation learning. A suggestion mining component can empower both, public and private sectors, to extract and leverage suggestions which are constantly expressed on various online platforms like Twitter[2] , TripAdvisor, or Reddit[3] for developing innovative services and products.

2 Task Definition

The early rule based approaches towards suggestion mining assumed that suggestions are always expressed using standard expressions like 'I recommend', 'I suggest that', 'You should', and created small evaluation datasets which were labeled in-house (Brun and Hagege, 2013; Ramanand et al., 2010). Only two of the previous studies released training datasets, which cover travel discussion forums (Wicaksono and Myaeng, 2013) and microblogs (Dong et al., 2013), while the review datasets from the previous works remain proprietary (Ramanand et al., 2010; Moghaddam, 2015). In our recent work, we perform a qualitative analysis of datasets from different sources, which includes investigation of linguistic properties of suggestions, relationship between sentiments and suggestions, and a laymans perception of suggestions (Negi, 2019). We also observed a low inter-annotator agreement in labeling sentences as suggestions and non-suggestions, and formulate a typology for sentences in context to suggestion detection, and design an annotation procedure based on this typology (Negi et al., 2018; Negi, 2019). For this shared task, we extend datasets from our previous studies, following the same task description and annotation method.

The Oxford dictionary defines suggestion as, *An idea or plan put forward for consideration*, and some of the listed synonyms of suggestions are *proposal, proposition, recommendation, advice, hint, tip, clue*. Many linguistic studies define how suggestions should be expressed in a standard use of language. However, in the context of text mining, we are dealing with user generated text on the web, which can be associated with multiple contexts, like the end user, domain etc. We observed in our layman annotation study, context may affect an annotator's judgment. In the absence of context, different annotators associate different contexts to a candidate sentence. We observed that the following concepts form an integral part of defining a suggestion in the context of suggestion mining and proposed an empirically driven and context-based definition of suggestions.

- **Surface structure:** Different surface structures (Chomsky, 1957; Crystal, 2011) can be used to express the underlying intention of giving the same suggestion. For example, *The nearby food outlets serve fresh local breakfast and are also cheaper* and *You can also have breakfast at the nearby food outlets which are cheaper and equally good.*

- **Context:** When dealing with specific use cases, context plays an important role in distinguishing a suggestion from a non-suggestion. Context may be present within

[2]https://www.twitter.com
[3]https://www.reddit.com

a given sentence. It can be a set of values corresponding to different variables that are provided explicitly and in addition to a given sentence. One or more of the following variables can constitute the context:

Domain: In this work, we refer to the source of a text as *domain*, which should not be considered in-line with the standard definition of domain. For example, in this shared task, we used hotel and suggestion forum domains.

Source text: The text in the entire source document to which a sentence belongs may also serve as a context, giving an insight into the discourse where the suggestion appeared.

Application or use case: Suggestions may sometimes be sought only around a specific topic, for example, room tips from hotel reviews. Suggestions can also be selectively mined for a certain class of users, for example, suggestions for future customers. All non-relevant suggestions in the data may be regarded as non-suggestions in this case.

Given that

- s denotes the surface structure of a sentence,
- C denotes additional context provided with s, where the context can be a set of values corresponding to certain variables, and
- $a(s, C)$ denotes the annotation agreement for the sentence, and t denotes a threshold value for the annotation agreement,

we write $S(s, C)$ to denote the *suggestion function*, which is defined as

$$S(s,C) = \begin{cases} Suggestion, & \text{if } a(s,C) \geq t \\ Non\text{-}suggestion, & \text{if } a(s,C) < t. \end{cases} \quad (1)$$

Depending on the choice of C, and, hence, on the value of $a(s, C)$, we identify four categories of sentences that a suggestion mining system is likely to encounter.

Explicit suggestions. *Explicit suggestions* are sentences for which S always outputs *Suggestion*, whether C is the empty set or not. They are like the *direct* and *conventionalised* forms of suggestions as defined by (Martínez Flor, 2005). It may also be the case that such sentences have a strong presence of context within their surface form, as in illustrated by *If you do end up here, be sure to specify a room at the back of the hotel.*

Explicit non-suggestions. These are the sentences for which S always outputs *Non-suggestion*, whether C is the empty set or not. For example, *Just returned from a 3 night stay.*

Implicit suggestions. These are sentences for which S outputs *Non-suggestion* only when C is the empty set. Typically, implicit suggestions do not posses the surface form of suggestions but the additional context helps the readers to identify them as suggestions. For example, *There is a parking garage on the corner of Forbes, so its pretty convenient* is labeled as a suggestion by the annotators when the context is revealed as that of a restaurant review. A sentence such as *Malahide is a pleasant village-turned-dormitory-town near the airport* can be considered as a suggestion given that it is obtained from a travel discussion thread for Dublin. These kind of sentences are observed to have a lower inter annotator agreement than the above two categories.

Implicit non-suggestions. These are sentences for which S outputs *Suggestion* only when C is an empty set. Typically, an implicit non-suggestion posses the surface form of suggestions but the context leads readers to identify them as non-suggestions. Such sentences may contain sarcasm. Examples include *Do not advertise if you don't know how to cook* appearing in a restaurant review and *The iPod is a very easy to use MP3 player, and if you can't figure this out, you shouldn't even own one* appearing in a MP3 player review.

The proposed categories provide the flexibility to change the scope of classes in a well defined manner, as well as to define context as per the application and use case. Based on the above four categoriese can define the scope of suggestion and non-suggestion classes for suggestion mining tasks. For open domain and cross domain suggestion mining, we proposed to limit the scope of suggestions to the *explicit suggestions*. Therefore, we set the definition of suggestion for this shared task as:

Let s be a sentence. If s is an explicit suggestion, assign the label *Suggestion*. Otherwise, assign the label *Non-suggestion*.

3 Dataset Annotation

A two phase annotation methodology, as proposed in our previous works (Negi et al., 2018; Negi, 2019) is followed.

3.1 Phase 1: Crowdsourced Annotations

This phase is performed using paid crowdsourcing, where each sentence is annotated by multiple layman annotators, and the set of annotators do not necessarily remain the same for all the sentences. We used Figure Eight[4] to collect layman annotations.

Annotators were also provided with the context, i.e. source text from where the sentence is extracted. They were simply asked to choose to label a sentence as suggestions if it contained expressions of suggestion, advice, tip, and recommendation. We aimed to collect implicit and explicit suggestions in this phase.

For quality control, before being allowed to perform a job, the annotators were presented with a set of test sentences which are similar to the actual questions except that their answers have already been provided by us to the system. We also submitted the explanation behind the correct answer. This way the test questions serve two purposes: test the annotators competency and understanding of the job, and train the annotator for the job. Crowdflower recommends certain best practices to prepare effective test questions.[5]. We submitted 30 test questions for each dataset. Each starting annotator was presented with 10 test questions, and only the annotators achieving an accuracy of 70% or more were allowed to proceed with the job. If an annotator passed the test and started the job, the remaining unseen test questions were presented to them in between the regular sentences without being notified. One sentence out of every 8 was a hidden test question. The accuracy score of a contributor on test questions is referred to as *Trust score* in a job. If an annotator's trust score dropped below a certain threshold during the course of the annotation, the system did not allow them to proceed further with the job. This threshold score was set to 70% in our case.

In addition to the hidden test questions, a minimum time for each annotator to stay on one page of the job was set. We set this time to 40 seconds (5 seconds on average for each sentence). If annotators appeared to be faster than that, they were automatically removed from the job. We restricted access to annotators from countries where English is a popular language and that are also likely to have a large crowdsourcing workforce. Most of the annotators came from Australia, Canada, Germany, India, Ireland, the United Kingdom, and the USA.

Annotation agreement: Crowdflower's *confidence* score describes the level of agreement between multiple contributors and the confidence in the validity of the result at the same time, we used a threshold confidence score of 0.6. However, it can be the case that a sentence is very ambiguous and cannot achieve the confidence score even after a large number of workers answered it. A maximum limit to the number of annotators is set in such case, and no further judgements are collected even if the threshold confidence is not reached. We set this limit to 5 annotators. Sentences that do not pass the confidence threshold of 0.6 are not included in the dataset.

3.2 Phase 2: Expert Annotations

This phase is performed by two in-house expert annotators, who are provided with the detailed annotation guidelines as compared to the phase 1 annotation guidelines, and the annotators are familiar with the problem definition and the task at hand. However, the annotators are not provided with the source text in this case. Phase 2 of the annotation is only applied to sentences that were labeled as suggestions in Phase 1, which drastically reduces the number of annotations to be performed in Phase 2.

Annotation Agreement: The inter-annotator agreement for Phase 2 was calculated by having two annotators label a subset of sentences for each domain (50 sentences). Cohen's kappa coefficient was used to measure the inter-annotator agreement. The remainder of the data instances were annotated by only one annotator. The fol-

[4]Earlier known as Crowdflower. https://www.figure-eight.com/

[5]https://success.crowdflower.com/hc/en-us/articles/213078963-Test-Question-Best-Practices

Subtask	Domain	Suggestion/Non-suggestion			IA agreement (phase 2)
		Training	Trial Test	Test	
A	Software developer suggestion forums (Uservoice)	1428 / 4296	296 / 742	87/746	0.81
B	Hotel reviews (Trip Advisor)	448 / 7086	404 / 3000	348/476	0.86

Table 2: Details of released datasets

lowing guidelines were provided to the annotators in Phase 2 :

- The intent of giving a suggestion and the suggested action or recommended entity should be explicitly stated in the sentence. *Try the cup cakes at the bakery next door* is a positive example. Other explicit forms of this suggestion could be: *I recommend the cup cakes at the bakery next door* or *You should definitely taste the cup cakes from the bakery next door*. An implicit way of expressing the suggestion could be *The cup cakes from the bakery next door were delicious*.

- The suggestion should have the intent of benefiting a stakeholder and should not be mere sarcasm or a joke. For example, *If the player doesn't work now, you can run it over with your car* would not pass this test.

Following are some of the scenarios of conflicting judgments observed in this phase of annotation:

- In the case of suggestion forums for specific domains, like a software developer forum, domain knowledge is required to distinguish an implicit non-suggestion from an explicit suggestion. Consider, for example, the two sentences, *It needs to be an integrated part of the phones functionality, that is why I put it in Framework* and *Secondly, you need to limit the number of apps that a publisher can submit with a particular key word*. The first sentence is a description of already existing functionality and is a context sentence in the original post, while the second is suggestion for a new feature.

- No concrete mention of what is being advised such as in *It'd be great if you would work on a solution to improve the situation*.

- At times, there was a confusion between information (fact) or suggestion (opinion). For example, *You can get a ticket that covers 6 of the National Gallery sites for only about US$10*.

In the final dataset, the sentences that are labeled as suggestions in Phase 2 of the annotation process are labeled as suggestions, while all other sentences are labeled as non-suggestions.

4 SemEval 2019 Shared Task

This is the pilot shared task on suggestion mining, the task is set as a binary sentence classification task, where the classes are suggestion and non-suggestion. As explained previously, explicit suggestions are deemed as the suggestion class, and rest of the sentences are considered as non-suggestions. The task is further split into two subtasks, named as A and B. Participating teams were to participate in at-least one of the two subtasks.

Datasets: Table 2 lists the details of the currently datasets released under this task and the inter-annotator agreement in the phase 2 of annotations. The class distribution is retained as obtained from a random sample of the source dataset used for annotation.
Software suggestion forum: The sentences for this dataset were scraped from the Uservoice platform[6]. Uservoice provides customer engagement tools tobrands, and therefore hosts dedicated suggestion forums for certain prod-ucts. The Feedly mobile application forum and the Windows developer forum are openly accessible. A sample of posts were scraped and split into sentences using the Stanford CoreNLP toolkit. Many suggestions are in the form of requests, which is less frequent in other domains. The text contains highly technical vocabulary related to the software which is being discussed.
Hotel reviews: Wachsmuth et al. (2014) provide a large dataset of hotel reviews collected from the TripAdvisor website[7]. They segmented the

[6]https://www.uservoice.com/
[7]https://www.tripadvisor.com/

reviews into statements so that each statement has only one sentiment label and have manually labeled the sentiments. Statements are equivalent to sentences, and comprise of one or more clauses. We further annotated these segments as *suggestion* and *non-suggestion*.

Sub-Task A: Train and test dataset belong to the same domain. The provided domain is suggestion forum sentences for software developers. Title of the posts are excluded, which are at times summary of the suggestion.
Sub-Task B: No training dataset is provided, and the test dataset belongs to a different domain than the subtask A, i.e. hotel reviews. The participants could use the training dataset from subtask A. Participants were not allowed to use the trial test set for subtask B as a training dataset, however they were allowed to use trial test set as a validation dataset.
Additional resources: Participants were allowed to use additional language resources, with one exception. Participants will be prohibited from using additional hand labeled training datasets for any of the domain.
Evaluation Metrics: Classification performance of the submitted systems if evaluated on the basis of F-1 score for the positive class, i.e. the *suggestion* class, which ranges from 0 to 1.
Precision suggestion (P_{sugg}): The fraction of instances which are actually suggestions out of the ones which are predicted as suggestions.
P_{sugg} = True Positives / (True Positives + False Positives)

Recall suggestion (R_{sugg}): The fraction of suggestion class instances which are correctly identified out of the total number of suggestions.
R_{sugg} = True Positives / (True Positives + False Negatives)

F1 score for the suggestion class is:
$F1_{sugg} = 2 * (P_{sugg} * R_{sugg}) / (P_{sugg} + R_{sugg})$

Baseline System A rule based classifier is employed using the existing rules from some of the related works, the rules which were dependent on the domain specific variables were excluded from the baseline. Table 4 provides the rules used in the baseline system.

Trial vs Test phase: A trial test dataset was released prior to the final test/evaluation dataset. The class distribution in the trial set was deliberately balanced in order to not bias the participants towards a specific class distribution for the evaluation phase, and keep the class distribution of trial set different from that of the final test set. This was because the trial test dataset labels were released prior to the final evalaution phase, and it was used as a validation dataset by the participants.

5 Participating Systems

A total of 33 teams participated in the evaluation phase, where all teams participated in the subtask A, and 16 of these also participated in subtask B. This number is lower than the trial phase submissions, where a total of 50 teams submitted their results on trial test dataset. Out of 33, 20 teams also submitted their system description papers. A summary of these 20 systems is provided in Table 3, listing results and corresponding methods. The highest F-score achieved was reasonably high i.e. 0.78 for subtask A, given a very low number of suggestion sentences in the test dataset 2. The highest F-score for subtask B was 0.858, where the ratio of suggestion and non-suggestion sentences in the test set was higher than subtask A.

Top 3 systems: BERT (Devlin et al., 2018) pre-trained language model remains the common method in the top three systems submitted in subtask A, which is one of the state of the art statistical language models. However, the most interesting results are provided by the best performing system in subtask B, which uses a rule based classifier, where rules comprise of both words and POS tags. The devised rule-based classifier (Potamias et al., 2019) assigns confidence scores to sentences on the basis of lexical patterns organised in pre-specified categories and lexical lists corresponding to each subtask. This rule based system also performed fairly well in subtask A, where it achieved rank 5.

Transfer and Unsupervised Learning: While a variety of pre-trained word embeddings and language models were employed, BERT remains the most popular means of transfer learning in the submitted systems, where 7 out of top 13 systems for subtask A used BERT.
For subtask B, only two systems used additional

Rank		Team Name	F-score		Method Used
Subtask A	Subtask B		Subtask A	Subtask B	
1	2	OleNet@Baidu (Jiaxiang et al., 2019)	0.7812	0.8579	Ensemble classifier (Logistic, GRU, FFA, CNN), with BERT
2	12	ThisIsCompetition (Park et al., 2019)	0.7778	0.6486	Ensemble classifier. Attention sentence encoder. BERT, CNN based word encoder.
3	5	m_y (Yamamoto and Sekiya, 2019)	0.7761	0.793	Distant supervision on unlabeled hotel reviews. BERT, ULMfit
4	NA	Yimmon (Zhuang, 2019)	0.7629	NA	Customised network, combination of convolution, self-attention and feed-forward layers. BERT
5	1	NTUA-ISLab (Potamias et al., 2019)	0.7488	0.858	Automatically learned rules
6	13	YNU-HPCC (Ping et al., 2019)	0.735	0.503	Ensemble classifier CNN, BILSTM and GRU. BERT
7	4	DS (Cabanski, 2019)	0.7273	0.8187	Ensemble classifier CNN and LSTM. BERT
9	NA	ZQM (Zhou et al., 2019)	0.715	NA	CNN. BERT
10	NA	MIDAS (Anand et al., 2019)	0.7011		Naive Bayes, Logistic Regression, SVM, LSTM. ULMFit
12	11	NL-FIIT (Pecar et al., 2019)	0.6816	0.685	Bi-LSTM. ELmO
13	3	Zoho (Prasanna and Seelan, 2019)	0.6807	0.8194	CNN. GloVe, BERT
14	NA	Lijunyi (Li and Ding, 2019)	0.6776	NA	Ensemble classifier, LSTM (attention-based), TextCNN, C-LSTM, Bi-LSTM. Word2Vec
19	7	WUT (Klimaszewski and Andruszkiewicz, 2019)	0.6293	0.7778	Domain-Adversarial Neural Networks (DANN). ELMo
23	6	Taurus (Oostdijk and Halteren, 2019)	0.5845	0.7925	Rules
27	NA	YNU_DYX (Ding et al., 2019)	0.5659	NA	BiLSTM, LSTM. Word2Vec, GloVe
28	9	INRIA (Markov and De la Clergerie, 2019)	0.5118	0.733	SVM, Logistic Regression. Hand crafted features.
29	17	SSN-SPARKS (S et al., 2019)	0.494	0.155	MultiLayer Perceptron, Random Forest and Convolutional Neural Network
30	14	DBMS-KU (Fatyanosa et al., 2019)	0.473	0.369	SVM, Linear Regression, Naive Bayes, CNN. GloVe
31	NA	UOL Artificial Intelligence Research Group (Ahmed et al., 2019)	0.3537	NA	Containment similarity, maximum common subgraph, Tree-based Pipeline Optimization Tool (TPOT)
NA	8	Hybrid RNN (Ezen-Can and F. Can, 2019)	NA	0.7449	Rule-based patterns, Glove, Bi-LSTM
32	10	Baseline	0.268	0.7329	Manually observed rules

Table 3: A summary of systems which are available as system description papers.

Keywords and phrases
needs to, need to
suggest, recommend, if, i wish, go for,should have, would, could have been
i would like, i'd like, i would love, I'd love, love to see
there should be, I wish, allow us to
Syntactic clues
If a modal verb or a base form of verb is present in the sentence. Eg, *I would prefer the unit to have a simple on off* switch.

Table 4: Rules for the baseline system

domain specific unlabeled data, i.e. hotel reviews, and were ranked as 3 and 5. All other submissions for subtask B relied on pre-trained word embeddings and language models.

Class Imbalance: Given that there was a major difference in the class distribution between training, trial test, and final test datasets, a minority of the top ten systems explicitly handle class imbalance by methods like oversampling (team MIDAS) and assigning weights to the predicted probability which are in proportion to the class distribution of the training data (team Yimmon). Other top 10 systems performed fairly well without any additional configuration for class imbalance in their classifiers.

Types of Classifiers: All systems except two used statistical classifiers, with most of them using neural network classifiers. Classifier ensembles also remain a favoured approach among the top ten systems. The neural network classifiers clearly outperformed SVM, Naive Bayes and Logistic regression. For subtask B, rule based classifier seem to do fairly well. The state of the art deep learning classifiers achieved a similar performance without any manual feature engineering, as compared to the carefully hand crafted rules.

6 Summary

We organised the pilot shared task on suggestion mining, which was framed as a binary text classification task, with two subtasks representing domain dependent and cross-domain/open domain evaluation. The task achieved a high level of participation, and most importantly a wide coverage in terms of methods and algorithms. The approaches covered automatically learned rule, carefully crafted linguistic features and rules, SOTA neural network classifiers, and SOTA transfer learning approaches. This shared task acted as a catalyst in pushing forward the state of the art for Suggestion Mining which otherwise received We plan to extend the task in future years with larger datasets, and the problem framed as the extraction of suggestion sentences from source texts in place of sentence classification. The problem definition here a better availability of document level context as compared to the sentence level context.

References

Usman Ahmed, Humera Liaquat, Luqman Ahmed, and Syed Jawad Hussain. 2019. Suggestion miner at semeval-2019 task 9: Suggestion detection in online forum using word graph. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019).*

Sarthak Anand, Debanjan Mahata, Kartik Aggarwal, Laiba Mehnaz, Simra Shahid, Haimin Zhang, Yaman Kumar, Rajiv Ratn Shah, and Karan Uppal. 2019. Midas at semeval-2019 task 9: Suggestion mining from online reviews using ulmfit. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019).*

Caroline Brun and Caroline Hagege. 2013. Suggestion mining: Detecting suggestions for improvement in users comments. *Research in Computing Science*.

Tobias Cabanski. 2019. Ds at semeval-2019 task 9: From suggestion mining with neural networks to adversarial cross-domain classification. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Noam Chomsky. 1957. *Syntactic Structures*. Mouton and Co., The Hague.

D. Crystal. 2011. *A Dictionary of Linguistics and Phonetics*. The Language Library. Wiley.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yunxia Ding, Xiaobing Zhou, and Xuejie Zhang. 2019. Ynu_dyx at semeval-2019 task 9: A stacked bilstm model for suggestion mining classific. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019).*

Li Dong, Furu Wei, Yajuan Duan, Xiaohua Liu, Ming Zhou, and Ke Xu. 2013. The automated acquisition of suggestions from tweets. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*. AAAI Press.

Aysu Ezen-Can and Ethem F. Can. 2019. Hybrid rnn at semeval-2019 task 9: Blending information sources for domain-independent suggestion mining. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Tirana Noor Fatyanosa, Al Hafiz Akbar, Maulana Siagian, and Masayoshi Aritsugi. 2019. Dbms-ku at semeval-2019 task 9: Exploring machine learning approaches in classifying text as suggestion or non-suggestion. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Liu Jiaxiang, Wang Shuohuan, and Sun Yu. 2019. Olenet at semeval-2019 task 9: Bert based multi-perspective models for suggestion mining. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Mateusz Klimaszewski and Piotr Andruszkiewicz. 2019. Wut at semeval-2019 task 9: Domain-adversarial neural networks for domain adaptation in suggestion mining. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Junyi Li and Haiyan Ding. 2019. Lijunyi at semeval-2019 task 9: An attention-based lstm model and ensemble of different models for suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Ilia Markov and Eric Villemonte De la Clergerie. 2019. Inria at semeval-2019 task 9: Suggestion mining using svm with handcrafted features. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Alicia Martínez Flor. 2005. A theoretical review of the speech act of suggesting: Towards a taxonomy for its use in flt. *Revista alicantina de estudios ingleses, No. 18 (Nov. 2005); pp. 167-187*.

Samaneh Moghaddam. 2015. Beyond sentiment analysis: mining defects and improvements from customer feedback. In *European Conference on Information Retrieval*, pages 400–410. Springer.

Saif M Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. pages 31–41.

Sapna Negi. 2016. Suggestion mining from opinionated text. In Alberto Pozzi, Elisabetta Fersini, Enza Messina, and Bing Liu, editors, *Sentiment Analysis in Social Networks*, chapter 8. Elsevier.

Sapna Negi. 2019. *Suggestion mining from text*. Ph.D. thesis, NUI Galway.

Sapna Negi, Kartik Asooja, Shubham Mehrotra, and Paul Buitelaar. 2016. A study of suggestions in opinionated texts and their automatic detection. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 170–178. Association for Computational Linguistics.

Sapna Negi and Paul Buitelaar. 2015. Towards the extraction of customer-to-customer suggestions from reviews. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2159–2167, Lisbon,Portugal. Association for Computational Linguistics.

Sapna Negi, Maarten de Rijke, and Paul Buitelaar. 2018. Open domain suggestion mining: Problem definition and datasets. *arXiv preprint arXiv:1806.02179*.

Nelleke Oostdijk and Hans van Halteren. 2019. Team taurus at semeval-2019 task 9: Expert-informed pattern recognition for suggestion mining. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Cheoneum Park, Juae Kim, Hyeon-gu Lee, Reinald Kim Amplayo, Harksoo Kim, Jungyun Seo, and Changki Lee. 2019. Thisiscompetition at semeval-2019 task 9: Bert is unstable for out-of-domain samples. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Samuel Pecar, Marian Simko, and Maria Bielikova. 2019. Nl-fiit at semeval-2019 task 9: Neural model ensemble for suggestion mining. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Yue Ping, Jin Wang, and Xuejie Zhang. 2019. Ynu-hpcc at semeval-2019 task 9: Using a bert and cnn-bilstm-gru model for suggestion mining. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Rolandos Alexandros Potamias, Alexandros Neofytou, and Georgios Siolas. 2019. Ntua-islab at semeval-2019 task 9: Mining suggestions in the wild. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Sai Prasanna and Sri Ananda Seelan. 2019. Zoho at semeval-2019 task 9: Semi-supervised domain adaptation using tri-training for suggestion mining. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

J Ramanand, Krishna Bhavsar, and Niranjan Pedanekar. 2010. Wishful thinking - finding suggestions and 'buy' wishes from product reviews. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 54–61. Association for Computational Linguistics.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. Semeval-2015 task 10: Sentiment analysis in twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 451–463, Denver, Colorado. Association for Computational Linguistics.

Rajalakshmi S, Angel Deborah S, S Milton Rajendram, and Mirnalinee T T. 2019. Ssn-sparks at semeval-2019 task 9: Mining suggestions from online reviews using deep learning techniques on augmented data. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019).*

Henning Wachsmuth, Martin Trenkmann, Benno Stein, Gregor Engels, and Tsvetomira Palakarska. 2014. A review corpus for argumentation analysis. In *Proceedings of the 15th International Conference on Computational Linguistics and Intelligent Text Processing*, volume 8404 of *LNCS*, pages 115–127, Kathmandu, Nepal. Springer.

Alfan Farizki Wicaksono and Sung-Hyon Myaeng. 2012. Mining advices from weblogs. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 2347–2350. ACM.

Alfan Farizki Wicaksono and Sung-Hyon Myaeng. 2013. Automatic extraction of advice-revealing sentences for advice mining from online forums. In *Proceedings of the Seventh International Conference on Knowledge Capture*, K-CAP '13, pages 97–104. ACM.

Masahiro Yamamoto and Toshiyuki Sekiya. 2019. m_y at semeval2019 task 9: Exploring bert for suggestion mining. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019).*

Qimin Zhou, Zhengxin Zhang, Hao Wu, and Linmao Wang. 2019. Zqm at semeval-2019 task9: A single layer cnn based on pre-trained model for suggestion mining. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Yimeng Zhuang. 2019. Yimmon at semeval-2019 task 9: Suggestion mining with hybrid augmented approaches. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019).*

m_y at SemEval-2019 Task 9: Exploring BERT for Suggestion Mining

Masahiro Yamamoto
Sony Corporation, Tokyo, Japan.
Masahiro.A.Yamamoto@sony.com

Toshiyuki Sekiya
Sony Corporation, Tokyo, Japan.
Toshiyuki.Sekiya@sony.com

Abstract

This paper presents our system to the SemEval-2019 Task 9, Suggestion Mining from Online Reviews and Forums. The goal of this task is to extract suggestions such as the expressions of tips, advice, and recommendations. We explore Bidirectional Encoder Representations from Transformers (BERT) focusing on target domain pre-training in Subtask A which provides training and test datasets in the same domain. In Subtask B, the cross domain suggestion mining task, we apply the idea of distant supervision. Our system obtained the third place in Subtask A and the fifth place in Subtask B, which demonstrates its efficacy of our approaches.

1 Introduction

In SemEval2019 Task 9, participants are required to build a model which can classify given sentences into suggestion and non-suggestion classes. We participate in two sub-tasks: domain specific suggestion mining task (Subtask A[1]) and cross domain suggestion mining task (Subtask B[2]). In Subtask A, the test dataset belongs to the same domain as the training and development datasets. These datasets are extracted from the suggestion forum for windows platform. In Subtask B, training and test datasets belong to separate domains. More specifically, the domain of the training dataset is entries from the windows forum and that of the test dataset is hotel reviews. Example sentences used in these tasks are listed in Table 1. For a description of these tasks please refer to (Negi et al., 2019).

Subtask A can be viewed as a binary text classification task. Recently, pre-training models, such as OpenAI GPI (Radford et al., 2018) and BERT (Devlin et al., 2018), have gained much attention with their ability to improve a number of downstream tasks. These models are pre-trained using unlabeled corpora and then fine-tuned on labeled datasets. We apply BERT to this task because it has achieved state-of-the-art performance in several text classification tasks. The difference to the original BERT model is that we further pre-train BERT model using an unlabeled corpus related to this domain. More concretely, we extract documents from a windows forum and run additional steps of pre-training using these documents, starting from the pre-trained BERT model.

Subtask A (windows platform)
P: xbox dev mode companion work ipv6 please...
N: I do not want to convert MSIs.
Subtask B (hotel review)
P: If you can, upgrade to an Ocean Front Room.
N: it doesn't have a very clean look.

Table 1: Example sentences in the test dataset. **P** means a positive sentence, i.e. suggestion sentence and **N** denotes a negative sentence, i.e. non-suggestion sentence.

In Subtask B, we apply the idea of distant supervision which has been firstly proposed by (Mintz et al., 2009). Distant supervision is a weakly supervised learning framework which tries to automatically generate noisy training examples. Specifically, we use the rule based system which is provided by the task organizer for creating a noisy training dataset and train the model based on them.

Our system significantly outperforms baseline methods on two subtasks. These results demonstrate its efficacy of our approaches, target domain pre-training in Subtask A and distant supervision in Subtask B.

[1]https://competitions.codalab.org/competitions/19955
[2]https://competitions.codalab.org/competitions/19956

2 System Description

Our model is built using a recent development of pre-training model, BERT. In the following, we describe details of this model first and then explain our systems in Subtask A and Subtask B.

2.1 BERT

BERT, proposed by (Devlin et al., 2018), has been shown to improve several tasks such as sentiment classification, calculating semantic textual similarity task, and recognizing textual entailment task. This model consists of several Transformer models (Vaswani et al., 2017) whose parameters are pre-trained on unlabeled corpora, Wikipedia and BooksCorpus (Zhu et al., 2015). Pre-training consists of two tasks, masked language modeling and next sentence prediction, and trained models of these unsupervised tasks are available.[3]

For the classification task, we added one layer to output predictions, suggestion or non-suggestion. These model parameters were then fine-tuned based on the labeled training dataset.

2.2 Subtask A

Subtask A is a classical document classification task, where training/development/test datasets consist of the same domain. The data has been collected from feedback posts on universal windows platform. We mainly use BERT in this task, however, in the following we describe several techniques to improve the score.

2.2.1 Target Domain Pre-training

The major difference between BERT and our system is the target domain pre-training. Typically, BERT training consists of two parts: pre-training on the general domain corpus and fine-tuning on the target task. On the other hand, our system consists of three training steps: pre-training on the general domain corpus, pre-training on the target domain corpus, and fine-tuning on the target task. More concretely, we further pre-train the model using a task specific unlabeled corpus scraped from the universal windows platform developer feedback site.[4] These documents are split into sentences and the model is then trained on two unsupervised tasks (i.e. masked language modeling and next sentence prediction). We should note here that we use officially provided pre-trained parameters as initial model parameters and further train the model based on target domain documents to obtain better network parameters.

This target domain pre-training can be considered as a similar framework of Universal Language Model Fine-Tuning (ULMFiT), proposed by (Howard and Ruder, 2018). In their framework, the model is firstly trained on general domain corpus to capture general features. In addition, the model is further trained on target task data to learn task specific features, and fine-tuned on the target task. This procedure is similar to our system. In other words, our work can be viewed as the extension of BERT model by using the idea of ULMFiT.

2.2.2 Model Averaging

(Reimers and Gurevych, 2017) showed that training deep neural network is sensitive to the initial weights and (Che et al., 2018) showed the effectiveness of ensembling models trained with different initialization. Furthermore, (Devlin et al., 2018) empirically showed that ensembling BERT models trained with different pre-training checkpoint leads to performance improvement. We follow this work and train three models with different pre-training checkpoint, then ensemble these models by simply averaging output scores.

2.3 Subtask B

Subtask B is cross-domain suggestion mining, where train and development datasets belong to windows platform domain, while the test dataset belongs to the hotel review domain. In this task, we do not use the datasets provided in Subtask A, because we find that models trained on these datasets tend to have poor performance (see also Sec. 3.2.2). We instead apply the paradigm of distant supervision.

Distant supervision is a framework to generate noisy annotated data automatically and use them as a training dataset. This idea has been firstly proposed by (Mintz et al., 2009) and well studied in the field of relation extraction (Riedel et al., 2010; Hoffmann et al., 2011).

2.3.1 Rule Based Labeling

For distant supervision, the initial labeling method is needed. In this work, we make use of the officially provided baseline method as an initial labeling tool. This system is based on a rule based

[3]https://github.com/google-research/bert

[4]https://wpdev.uservoice.com/forums/110705-universal-windows-platform

	Subtask A			Subtask B		
	Train	Dev	Test	Train	Dev	Test
Suggestions	2,085	296	87	-	404	348
Non Suggestions	6,415	296	746	-	404	476
All	8,500	592	833	-	808	824

Table 2: Statistics of datasets.

method. For example, if a specific word such as ”suggest” is in the sentence, then the sentence is predicted as a suggestion sentence. For more details, please see the code in the official repository.[5]

2.3.2 Model Training Procedure

After the labeled corpus is generated, we trained a model using this corpus. Here, we do not label all of the unlabeled sentences via the rule based system. Instead, we split the sentences into several pieces and label one of them by using the rule based system. A model is trained on the noisy labeled dataset and we label another piece through the trained model. We apply this procedure iteratively until the model is trained on the last piece. More specifically, our training procedure can be summarized as follows:

1. We prepare the unlabeled hotel review corpus and split it into N pieces (corpus C_1, C_2, C_3, ..., C_N).
2. We apply the baseline method which is provided by the task organizer to the corpus C_1 and treat predicted labels as true labels.
3. The model is then trained using the labeled corpus C'_1.
4. We apply the trained model to the corpus C_2 and treat predicted labels as true labels.
5. A new model is trained using the labeled corpus C'_2 and labels of sentences in C_3 are predicted by this model.
6. We iteratively apply the above procedure until the model is trained on the corpus C_N.

3 Experiments

Table 2 shows the statistics of datasets which are used in Subtask A and Subtask B. These datasets are available at the official repository.[6] For evaluating systems, F1 score for the positive class, i.e. the suggestion class, is employed.

[5]https://github.com/Semeval2019Task9/Subtask-B/blob/master/semeval-task9-baseline.py

[6]https://github.com/Semeval2019Task9

System	Dev F1	Test F1
Baseline	0.721	0.267
BERT BASE (Single)	0.845	0.731
BERT BASE (Sgl. + Tgt.)	0.866	0.755
BERT LARGE (Single)	0.867	0.737
BERT LARGE (Sgl. + Tgt.)	0.882	0.759
BERT LARGE (Ens. + Tgt.)	0.890	0.776

Table 3: Results of Subtask A. **Single** or **Sgl.** denotes the single model and **Ens.** means the ensembled models. **Tgt.** denotes the pre-training on the target domain corpus.

3.1 Subtask A

3.1.1 Settings

We employed the official BERT model. We used both BASE model which has 12 Transformer layers, 12 self-attention heads, and 768 hidden size, and LARGE model which has 24 Transformer layers, 16 self-attention heads, and 1024 hidden size. There are 110 million parameters in total in BASE model and 340 million parameters in LARGE model.

As for the target domain pre-training, we obtained the windows review corpus and split it into sentences using NLTK.[7] The number of scraped documents is 2,325 and we used these documents as the unlabeled corpus for further pre-training the BERT model.

3.1.2 Results and Discussions

Table 3 shows the results of the experiment. Here, the baseline method is a rule based method as explained in Sec. 2.3.1. From Table 3, we can conclude the following four things:

First, BERT LARGE model outperformed BERT BASE model despite of the small size of the dataset. In general, it has been known that increasing the model size leads to an improvement on large scale tasks such as machine translation, and this does not be applied to small scale tasks

[7]https://www.nltk.org/

except for (Devlin et al., 2018). They showed a similar tendency in another small scale task. These results demonstrate that large size models improve results not only on large scale tasks but also on small scale tasks, if the model has been well pre-trained.

Second, the effect of the target domain pre-training is not trivial. The improvement can be observed in both BASE and LARGE model. In BASE model, the F1 score is pushed from 0.845 to 0.866 in the development dataset and from 0.731 to 0.755 in the test dataset. In LARGE model, the score is improved from 0.867 to 0.882 in the development dataset and from 0.737 to 0.759 in the test dataset. These results show that better models are produced by target domain pre-training even if we do not have large, domain-specific documents.

Third, ensembling models leads to further performance improvement.

Fourth, Test F1 score is much lower than Dev F1 score. This has also been observed by other teams. In concrete, many teams have achieved over 0.850 F1 score on the development dataset, however, no team has achieved over 0.800 F1 score on the test dataset. It might have something to do with the small size of the development dataset size or the difference in the label distribution.

3.2 Subtask B

3.2.1 Settings

In Subtask B, we got unlabeled hotel review documents provided by (Wachsmuth et al., 2014).[8] We split documents into sentences using the NLTK package and randomly extracted 500,000 sentences. These sentences were further split into 5 pieces.

We firstly ran the baseline method and automatically labeled 100,000 sentences. These sentences were used as a training dataset to train the machine learning model. We used the BERT BASE model and set the default hyperparameters except for the training epochs. To avoid over-fitting, we trained the model for only one epoch.

As described in Sec. 2.3, another 100,000 sentences were automatically labeled using the trained model and we trained a new model using this labeled data. We iteratively applied this procedure and submitted results predicted by the final model.

[8]http://argumentation.bplaced.net/arguana/data

System	Dev F1	Test F1
Baseline	0.774	0.732
BERT BASE (windows)	0.308	0.419
BERT BASE (1st model)	0.800	0.785
BERT BASE (5th model)	0.817	0.793

Table 4: Results of Subtask B. **BERT BASE (windows)** is the model trained on windows corpus provided in Subtask A.

3.2.2 Results and Discussions

Table 4 shows the results of the experiment. As you can see in Table 4, BERT BASE (windows), the model trained on the other domain corpus, shows poor performance while the baseline method has achieved a much better score. This motivated us to apply the idea of distant supervision.

The model based on distant supervision significantly outperformed the baseline model. This result shows that the distant supervision idea can be applied successfully in the cross domain suggestion mining task. Furthermore, we can see that our iterative approach leads to more performance improvement (from 0.800 to 0.817 in the development dataset and 0.785 to 0.793 in the test dataset). We currently do not know why the F1 score has been improved, however, one interpretation could be that our iterative framework avoids over-fitting to the one model and learns more general decision boundaries. A detailed study of this effect is future work.

4 Conclusion

This paper explains our submission to SemEval2019 Task 9, Subtask A and Subtask B. We explored BERT models focusing on the target domain pre-training in Subtask A and the idea of the distant supervision in Subtask B. Our approach obtained the third place in Subtask A and the fifth place in Subtask B.

In the future, we will further investigate the effect of these approaches in other tasks.

References

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better ud parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. *arXiv preprint arXiv:1807.03121*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 328–339.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Alec Radford, Karthik Narasimhan, Time Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, OpenAI.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. *arXiv preprint arXiv:1707.09861*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, Gregor Engels, and Tsvetomira Palakarska. 2014. A review corpus for argumentation analysis. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 115–127. Springer.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

SemEval 2019 Task 10: Math Question Answering

Mark Hopkins
Department of Mathematics
Reed College
hopkinsm@reed.edu

Ronan Le Bras and **Cristian Petrescu-Prahova**
Allen Institute for Artificial Intelligence
{ronanlb,cristipp}@allenai.org

Gabriel Stanovsky and **Hannaneh Hajishirzi** and **Rik Koncel-Kedziorski**
Allen School of Computer Science and Engineering
University of Washington
{gabis, hannaneh,kedzior}@uw.edu

Abstract

We report on the SemEval 2019 task on math question answering. We provided a question set derived from Math SAT practice exams, including 2778 training questions and 1082 test questions. For a significant subset of these questions, we also provided SMT-LIB logical form annotations and an interpreter that could solve these logical forms. Systems were evaluated based on the percentage of correctly answered questions. The top system correctly answered 45% of the test questions, a considerable improvement over the 17% random guessing baseline.

1 Overview

Over the past four years, there has been a surge of interest in math question answering. Research groups from around the globe have published papers on the topic, including MIT (Kushman et al., 2014), University of Washington (Hosseini et al., 2014; Koncel-Kedziorski et al., 2015), National Institute of Informatics, Japan (Matsuzaki et al., 2014), University of Illinois (Roy and Roth, 2015), Microsoft Research (Shi et al., 2015; Upadhyay and Chang, 2016), Baidu (Zhou et al., 2015), Arizona State University (Mitra and Baral, 2016), KU Leuven (Dries et al., 2017), Carnegie Mellon University (Sachan et al., 2017), Tencent (Wang et al., 2017), and DeepMind (Ling et al., 2017).

Math question answering has several attractive properties that have rekindled this interest:

1. It is easy to evaluate. Usually there is a single correct answer for a given question, either numeric or multiple-choice.

2. In order to achieve robust results, systems require some (explicit or implicit) semantic representation of the question language. Consider the first question in Figure 1. The correct answer (11) has a subtle relationship to the other quantities mentioned in the text (3 and 15). There is no obvious shortcut (like word association metrics on a bag-of-words representation of the question) to guessing 11.

3. Math questions exhibit interesting semantic phenomena like cross-sentence coreference and indirect coreference (e.g. in the geometry question from Figure 1, "the third side" refers to a triangle introduced in a previous sentence).

Closed-vocabulary algebra: Suppose 3x + y = 15, where x is a positive integer. What is the difference between the largest possible value of y and the smallest possible value of x, assuming that y is also a positive integer?

Open-vocabulary algebra: At a basketball tournament involving 8 teams, each team played 4 games with each of the other teams. How many games were played at this tournament?

Geometry: The lengths of two sides of a triangle are $(x-2)$ and $(x+2)$, where $x > 2$. Which of the following ranges includes all and only the possible values of the third side y? (A) $0 < y < x$ (B) $0 < y < 2x$ (C) $4 < y < 2x$

Figure 1: Example math questions from different genres.

This task sought to unify the somewhat divergent research efforts and to address certain recognized data issues that have developed in the nascent

```
man buys an article for 10 % less than its value and sells it for 10 % more than its value . His gain or loss percent is ?
man can row upstream at 10 kmph and downstream at 20 kmph , and then find the speed of the man in still water ?
man can row upstream at 25 kmph and downstream at 35 kmph , and then find the speed of the man in still water ?
man can row upstream at 30 kmph and downstream at 40 kmph , and then find the speed of the man in still water ?
man is 15 years older than his son . In two years , his age will be twice the age of his son . The present age of his son is :
man is 20 years older than his son . In two years , his age will be twice the age of his son . The present age of his son is :
man is 21 years older than his son . In two years , his age will be twice the age of his son . The present age of his son is :
man is 24 years older than his son . In two years , his age will be twice the age of his son . The present age of his son is :
man is 28 years older than his son . In two years , his age will be twice the age of his son . The present age of his son is :
man is 36 years older than his son . In two years , his age will be twice the age of his son . The present age of his son is :
man walks at speed of 8 km / h 300 mtrs length train crosses man in 30 sec from back , find speed of train ?
maximun number of identical pieces ( of same size ) of a cake by making only 3 cuts ?
```

Figure 2: Sample of questions of the AQuA dataset.

	Closed Algebra	**Open Algebra**	**Geometry**	**Total**
# test questions	476	216	335	1082
# training questions	1068	353	701	2778

Table 1: Data resources for the three subtasks. Note that the fourth column ("Total") also includes a small minority of questions that do not fall into the three major categories.

phase of this subfield. We discuss these issues in the next section.

2 Existing Resources

Existing datasets are either scraped from the web and filtered (Kushman et al., 2014; Hosseini et al., 2014; Roy and Roth, 2015; Shi et al., 2015) or crowdsourced (Ling et al., 2017).

The scraped datasets have been observed to be narrow in semantic scope, and to exhibit considerable lexical overlap from question to question (Koncel-Kedziorski et al., 2015; Roy and Roth, 2016). Also, they tend to be curated to showcase proposed models (e.g. by requiring every quantity present in the semantic representation to be explicitly mentioned in the question).

DeepMind (Ling et al., 2017) provided a public, large-scale dataset called AQuA, consisting of approximately 100,000 questions. This dataset was created by using crowdsourcing to augment a nucleus of web-scraped questions. Unfortunately, the result is extremely redundant and noisy. Figure 2 shows a typical excerpt from the sorted list of AQuA questions. Note the amount of repeated boilerplate and the low quality of the language. While AQuA may prove useful for training, it is inappropriate as an evaluation set.

3 Resource: Train/Test Data from Math SAT practice tests

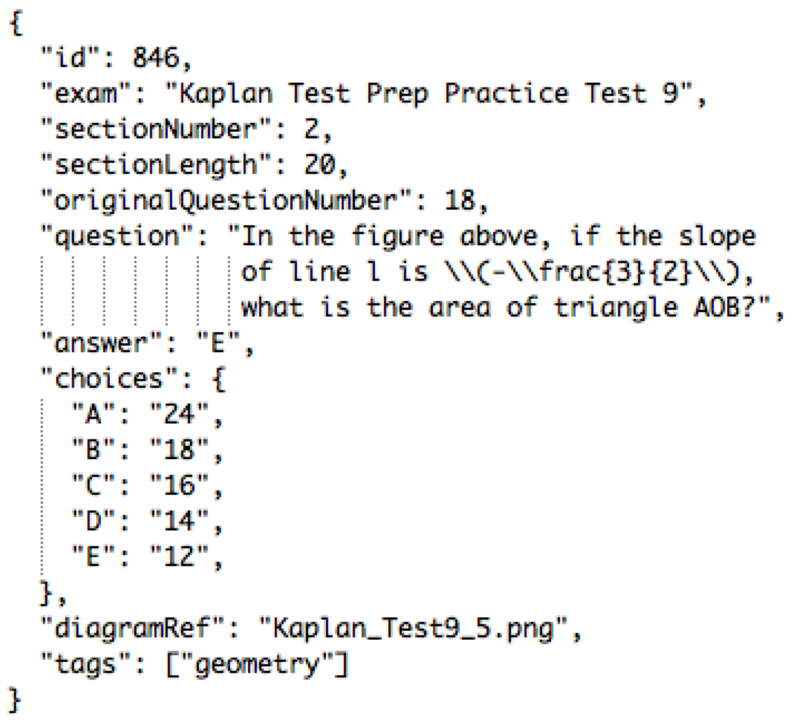
Over the course of the Euclid project (Hosseini et al., 2014; Seo et al., 2014, 2015; Koncel-Kedziorski et al., 2015; Hopkins et al., 2017) at the Allen Institute for Artificial Intelligence, we curated a sizable collection of practice exams for Math SAT study guides. These were originally used as training and test in a paper that appeared at EMNLP (Hopkins et al., 2017). At the time, this dataset consisted of 648 training questions (12 practice tests) and 1082 test questions (21 practice tests). For this task, we expanded the training set to include 2778 training questions (over 50 practice tests).

Because the data is a compilation of SAT practice exams in their entirety, its distribution of topics corresponds to (at least one authority's idea of) the breadth of knowledge expected of an incoming college student. It is curated by impartial third parties (the study guide publishers) and is thus not a priori biased towards any particular model. The language is high quality and diverse.

3.1 Data Collection Process and Format

First, practice exams were scanned from physical printed copies of SAT study guides by Kaplan, McGraw-Hill, and the Princeton Review, among others. We also processed official PDF practice exams issued by the College Board. Then, trained workers manually encoded each exam as a LaTeX document, attempting to preserve the original formatting as much as possible. PDFs generated from the LaTeX documents were compared against the original scans to ensure quality and corrections were made as necessary.

Finally, the LaTeX documents were automatically converted into the JSON format shown in Figure 3. Diagrams from the exams are stored as Portable Network Graphics (PNG) files in a com-



```
{
  "id": 846,
  "exam": "Kaplan Test Prep Practice Test 9",
  "sectionNumber": 2,
  "sectionLength": 20,
  "originalQuestionNumber": 18,
  "question": "In the figure above, if the slope
              of line l is \\(-\\frac{3}{2}\\),
              what is the area of triangle AOB?",
  "answer": "E",
  "choices": {
    "A": "24",
    "B": "18",
    "C": "16",
    "D": "14",
    "E": "12",
  },
  "diagramRef": "Kaplan_Test9_5.png",
  "tags": ["geometry"]
}
```

Figure 3: JSON representation of a math SAT question.

mon directory. If a question refers to a diagram, then this is captured by the "diagramRef" field. Not all questions are multiple-choice. For non-multiple choice questions, the "answer" field contains the answer itself (as a string) rather than the choice key.

3.2 Subtasks

The Math SAT contains three broadly discernible subcategories (examples of which are provided in Figure 1):

1. **Closed-vocabulary algebra (approximately 44% of the questions):** Algebra word problems described with a circumscribed mathematical vocabulary. Note that the language and semantics can still be quite involved (see the first example in Figure 1).

2. **Open-vocabulary algebra (approximately 20% of the questions):** Algebra word problems described with an open-ended vocabulary, often involving real-world situation descriptions.

3. **Geometry (approximately 31% of the questions):** In contrast to the algebra subdomains, these often involve diagrams and thus require methods that perform joint reasoning over language and images, e.g. (Seo et al., 2014).

As part of the digitization process described in the previous section, we have also tagged the questions based on this categorization. A small minority (approximately 5%) of questions do not fall into any of these categories.

This categorization provides the basis for three subtasks: (1) closed algebra QA, (2) open algebra QA, and (3) geometry QA. Note that the algebra subtasks do not involve diagrams (algebra questions involving diagrams are classified into the small "other" category). Table 1 shows the number of questions collected, organized by subtask.

4 Additional Resource: Logical Forms for Closed Algebra

To make it easier for researchers to build systems, we provided logical form annotations for a majority of the training questions in the closed algebra subtask, as well as an engine that solves these logical forms. The logical form language was introduced in (Hopkins et al., 2017). Figure 4 shows an example logical form for the question "The sum of a two-digit number and its reverse is 121. What is the number?" The logical form language adopts

```
(declare-const q Number)
(assert (= (Count (Digits q)) 2))
(assert (= 121 (Sum q (Reverse q))))
(assert (Query q))
```

Figure 4: Logical form annotation for the question "The sum of a two-digit number and its reverse is 121. What is the number?"

the popular SMT-LIB syntax (Barrett et al., 2017), to facilitate language expansion and the building of alternate engines.

As part of the SemEval task, we provided:

- Logical form annotations for 50% of the closed algebra training data.
- Thorough documentation of the logical form language.
- An interpreter that can solve all provided annotations.

The logical form interpreter is an extended version of the one described in (Hopkins et al., 2017). This interpreter is written in Scala. We provide Scala, Java, and Python APIs.

Note: the logical form annotations and interpreter were provided to help lower the barrier to entry, but participants were not required to use them.

5 Evaluation Methodology and Baseline

For each subtask, the main evaluation metric was simply question accuracy, i.e. the number of correctly answered questions. We provided a Python script that took as input a list of JSON datum { id: <id>, response: "<response>" }, where <id> is the integer index of a question and <response> is the guessed response (either a choice key or a numeric string). Its output was the number of correct responses divided by the total number of questions in the subtask.

While the main evaluation metric included no penalties for guessing, we also computed a secondary metric that implements the actual evaluation metric used to score these SATs. This metric is the number of correct questions, minus 1/4 point for each incorrect guess. We include this metric to challenge participants to investigate high-precision QA systems.

For each subtask, we provided a simple Python baseline that reads in a JSON file containing the evaluation questions, and randomly guesses a choice key for each multiple choice question, and "0" for each numeric-answer question.

To train their systems, participants were permitted to use the following public resources: (a) the provided SAT training data and annotations, (b) data collected in MAWPS (Koncel-Kedziorski et al., 2016), (c) AQuA. Participants were also welcome to use standard public corpora for training word vector representations, language models, etc.

6 Evaluation

Table 2 shows the teams (and their affiliations) that submitted systems that beat the baseline in at least one task. Tables 3, 4, 5, and 6 compares the performance of these systems. The AiFu systems (Ding et al., 2019) outperformed the other entries by a wide margin.

6.1 The AiFu System

The AiFu system (Ding et al., 2019), like other math question answering systems designed for complex domains (Shi et al., 2015; Hopkins et al., 2017), followed the architecture in Figure 5. First, a natural language question is transformed into a logical form via a "translator" (semantic parser), then this logical form is given to a symbolic solver (after a suitable format conversion).

Like the previous semantic parsing approaches, the AiFu semantic parser is engineered manually, using the training set as a guide.

AiFu also incorporates a neural network-based system trained to guess the answer to a multiple choice question based on its question word sequence. Although this system does not work well independently, it boosts the performance of the overall system when used as a fallback for questions that go unanswered by the symbolic system.

The AiFu team submitted two variants of their system, referred to in the comparison tables as AiFu 1 and AiFu 2.

6.2 The ProblemSolver System

The ProblemSolver system (Luo et al., 2019) combines two approaches.

The first approach is a neural sequence-to-sequence translator that maps a question, e.g. "If $x + 345 = 111$, what is the value of x", to a response, e.g. "-234". Because the provided data is insufficiently large for training the sequence-

Team Name	Affiliation	Citation
AiFu	iFLYTEK Research, Shanghai Research Center for Brain Science and Brain-Inspired Intelligence, Fudan University, Massey University, University of Science and Technology of China	(Ding et al., 2019)
ProblemSolver	University of Tuebingen, Germany	(Luo et al., 2019)
FAST	National University of Computer and Emerging Sciences, Pakistan	

Table 2: Teams whose entries exceeded baseline performance on at least one subtask. FAST (and the two anonymous systems) did not provide system description papers.

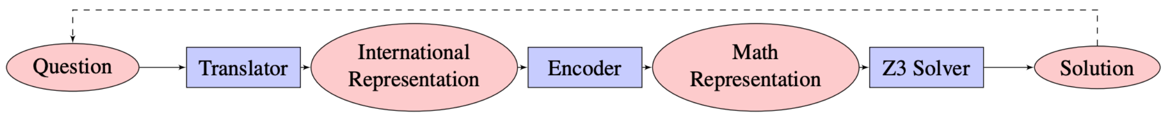


Figure 5: The architecture of AiFu. Figure taken from (Ding et al., 2019).

Team	Accuracy	Penalized
AiFu 1	.454 (1)	.368 (1)
AiFu 2	.376 (2)	.280 (2)
Anonymous 1	.208 (3)	.089 (3)
Anonymous 2	.196 (4)	.074 (4)
FAST	.173 (5)	.007 (6)
baseline	*.170* (6)	*.043* (5)
ProblemSolver	.149 (7)	-.021 (7)

Table 3: Results on the overall task (only showing entries that exceeded baseline performance on at least one subtask). System rank on each metric is shown in parentheticals.

Team	Accuracy	Penalized
AiFu 1	.706 (1)	.658 (1)
AiFu 2	.632 (2)	.576 (2)
Anonymous 2	.196 (3)	.075 (3)
Anonymous 1	.187 (4)	.064 (4)
FAST	.183 (5)	.020 (5)
ProblemSolver	.157 (6)	-.012 (7)
baseline	*.146* (7)	*.015* (6)

Table 4: Results on the closed-vocabulary algebra subtask (only showing entries that exceeded baseline performance on at least one subtask). System rank on each metric is shown in parentheticals.

Team	Accuracy	Penalized
AiFu 1	.251 (1)	.145 (1)
AiFu 2	.247 (2)	.140 (2)
Anonymous 2	.247 (2)	.140 (2)
Anonymous 1	.196 (4)	.079 (4)
baseline	*.174* (5)	*.052* (5)
FAST	.146 (6)	-.025 (6)
ProblemSolver	.146 (6)	-.025 (6)

Table 5: Results on the open-vocabulary algebra subtask (only showing entries that exceeded baseline performance on at least one subtask). System rank on each metric is shown in parentheticals.

Team	Accuracy	Penalized
AiFu 1	.265 (1)	.145 (1)
Anonymous 1	.216 (2)	.099 (2)
baseline	*.212* (3)	*.095* (3)
FAST	.159 (4)	-.009 (6)
Anonymous 2	.152 (5)	.023 (4)
ProblemSolver	.152 (6)	-.018 (7)
AiFu 2	.134 (7)	-.003 (5)

Table 6: Results on the geometry subtask (only showing entries that exceeded baseline performance on at least one subtask). System rank on each metric is shown in parentheticals.

to-sequence model, they use data augmentation methods to increase the data size to over 600K questions.

The second approach is an adaptation of the arithmetic tree approach of (Roy and Roth, 2015) to Math SAT question answering.

The combination of these techniques provides a minor improvement over the random guessing baseline.

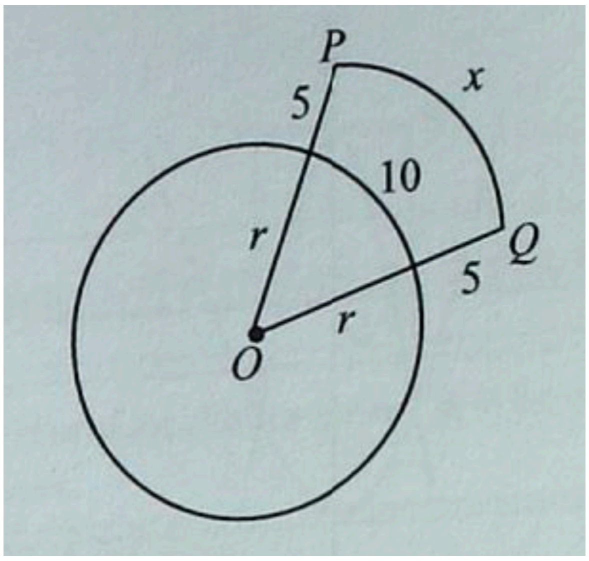


Figure 6: Example diagram accompanying the question "In the circle above, the length of an arc is 10, and there are two radii shown extended by 5 units outside the circle. If the length of arc Q is x, what must x equal?"

7 Discussion

Lately, math question answering seems to have bifurcated into two distinct approaches:

1. Simple, elegant machine learning approaches that work mainly on narrowly-scoped datasets with considerable redundancy.

2. Engineering-heavy, rule-based approaches that significantly outperform ML approaches on more realistic datasets, but are laborious to scale to new domains.

This SemEval task provides additional examples of these two approaches. As NLP researchers, we could focus on narrow-scope datasets for the time being[1], improving the performance of scalable ML approaches on these datasets. However the challenges presented by more difficult datasets, like the Math SAT data provided in this task, are intriguing and important. For instance:

How do we synthesize information that comes from heterogenous sources (e.g. from text and diagrams)?

Many of the geometry questions require a generalized notion of coreference resolution that spans language and vision. Figure 6 shows an example diagram that accompanies the question "In the circle above, the length of an arc is 10, and there are two radii shown extended by 5 units outside the circle. If the length of arc Q is x, what must x equal?". It remains an open question how to reliably resolve textual references like "the circle above" and "two radii shown" with diagram components. (Seo et al., 2014, 2015) provide a starting point for this area, but their dataset consisted of less than 100 diagrams. Hopefully our larger resource can help spur research into this research question.

How can machine learning be leveraged to reduce (or eliminate) the burden of engineering semantic parsers for complicated domains?

Given that the only techniques that have so far found success on Math SAT question answering (Hopkins et al., 2017; Ding et al., 2019) have involved semantic parsers with engineered rules, it suggests that one path forward might be to use machine learning to facilitate the engineering or elicitation of such rules for low-resource QA domains.

How do we create ML systems for diverse datasets for which we do not (and will never have) millions of training instances?

Despite the huge industry surrounding the Math SAT, it was still challenging to find and digitize over 50 distinct practice exams. Having millions of instances is not feasible. We argue that there will always be a long tail of domains for which we do not have millions of training instances, and we need ways to induce performant systems on this scale of data.

8 Conclusion

We have digitized over 70 SAT practice exams as a resource for driving research in math (and low-resource) question answering. We have also provided logical form annotations for approximately half of the closed-vocabulary algebra questions in the training data. The top system in our competition, AiFu, correctly answered 45% of the test questions, compared to a random guessing baseline of 17%. Our data and logical forms are available at https://github.com/allenai/semeval-2019-task-10, subject to the terms and conditions specified in that repository.

[1]Indeed, this seems to be the prevailing opinion of the hundredsome task participants who abandoned the task after obtaining the data.

References

Clark Barrett, Pascal Fontaine, and Cesare Tinelli. 2017. The SMT-LIB Standard: Version 2.6. Technical report, Department of Computer Science, The University of Iowa. Available at `www.SMT-LIB.org`.

Keyu Ding, Yifan Liu, Yi Zhou, Binbin Deng, Chaoyang Peng, Dinglong Xue, Qinzhuo Wu, Qi Zhang, and Enhong Chen. 2019. Aifu at semeval-2019 task 10: A symbolic and sub-symbolic integrated system for sat math question answering. In *SemEval-2019 Task 10*.

Anton Dries, Angelika Kimmig, Jesse Davis, Vaishak Belle, and Luc De Raedt. 2017. Solving probability problems in natural language. In *International Joint Conference on Artificial Intelligence*.

Mark Hopkins, Cristian Petrescu-Prahova, Roie Levin, Ronan Le Bras, Alvaro Herrasti, and Vidur Joshi. 2017. Beyond sentential semantic parsing: Tackling the math sat with a cascade of tree transducers. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 795–804.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *TACL*, 3:585–597.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. Mawps: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157.

Nate Kushman, Luke S. Zettlemoyer, Regina Barzilay, and Yoav Artzi. 2014. Learning to automatically solve algebra word problems. In *ACL*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

Xuefeng Luo, Alina Baranova, and Jonas Biegert. 2019. Problemsolver at semeval-2019 task 10: Sequence-to-sequence learning and expression trees. In *SemEval-2019 Task 10*.

Takuya Matsuzaki, Hidenao Iwane, Hirokazu Anai, and Noriko H Arai. 2014. The most uncreative examinee: A first step toward wide coverage natural language math problem solving. In *AAAI*, pages 1098–1104.

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *ACL*.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *EMNLP*.

Subhro Roy and Dan Roth. 2016. Unit dependency graph and its application to arithmetic word problem solving. *arXiv preprint arXiv:1612.00969*.

Mrinmaya Sachan, Avinava Dubey, and Eric P. Xing. 2017. From textbooks to knowledge: A case study in harvesting axiomatic knowledge from textbooks to solve geometry problems. In *EMNLP*.

Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, and Oren Etzioni. 2014. Diagram understanding in geometry questions. In *AAAI*.

Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving geometry problems: Combining text and diagram interpretation. In *EMNLP*.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *EMNLP*.

Shyam Upadhyay and Ming-Wei Chang. 2016. Annotating derivations: A new evaluation strategy and dataset for algebra word problems. *CoRR*, abs/1609.07197.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *EMNLP*.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *EMNLP*.

AiFu at SemEval-2019 Task 10: A Symbolic and Sub-symbolic Integrated System for SAT Math Question Answering

Keyu Ding[1,2,6], Yifan Liu[1,2], Yi Zhou[3,5], Binbin Deng[1,2], Chaoyang Peng[1,2],
Dinglong Xue[1,2], Qinzhuo Wu[3], Qi Zhang[3]
and Enhong Chen[5]

[1]iFLYTEK Research
[2]State Key Laboratory of Cognitive Intelligence, iFLYTEK, P.R. China
[3]Shanghai Research Center for Brain Science and Brain-Inspired Intelligence/ZhangJiang Lab
[4]School of Computer Science, Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
[5]School of Natural and Computational Sciences, Massey University
[6]University of Science and Technology of China
{kyding,yfliu7,bbdeng,zypeng,dlxue}@iflytek.com
yzhou@bsbii.cn
{qzwu17,qz}@fudan.edu.cn
cheneh@ustc.edu.cn

Abstract

`AiFu` has won the first place in the SemEval-2019 Task [10] - "Math Question Answering"(Hopkins et al.) competition. This paper is to describe how it works technically and to report and analyze some essential experimental results.

1 Introduction

Recently, math question answering has attracted a lot of attention in the AI community, both in academia and in industry (Matsuzaki et al., 2017) (Wang et al., 2017) (Huang et al., 2016) (Wang et al., 2018) (Hosseini et al., 2014)(Huang et al., 2018a) (Huang et al., 2018b) (Kushman et al., 2014) (Liang et al., 2017) (Mitra and Baral, 2016) (Zhou et al., 2015) (Roy and Roth, 2017)(Hopkins et al., 2017). On one side, it raises a difficult yet workable challenge for the current development of AI research. In order to tackle this challenge, one has to integrate and advance many subareas in AI including knowledge representation and reasoning, machine learning, natural language understanding and image understanding. On the other side, math question answering itself has important commercial value in the AI+Education industry.

Against this backdrop, SemEval-2019 organizes a competition on math question answering, namely Task 10 (Hopkins et al.). In this task, an opportunity is provided for Math Question-Answering systems to test themselves on a benchmark that consists of many math questions collected from the US Math Scholastic Achievement Test (SAT).

We have implemented a prototype system, called `AiFu`, to tackle this challenge, and it has won the first place at the end. `AiFu` is an integrated system that combines the state-of-the-art approaches from many important subareas in AI, and more importantly, it also develops and justifies some new ideas and techniques.

This paper is to describe how `AiFu` works technically and to report and analyze some essential experimental results. In the next section, we go through the technical details of `AiFu` that consists of many essential components including representation, reasoning and natural language understanding. In Section 3, we report our experimental results and shed new insights on why `AiFu` works in some cases but not in others. Finally, we conclude this paper and point out some future directions.

2 Method

Figure 1 depicts the overall architecture of `AiFu`. Given a mathematical question, a translator is used to convert it into its internal representation, which is sent to an encoder to further convert it into a math representation that can be directly used by the SMT solver Z3 (De Moura and Bjørner, 2008). Finally, by calling Z3, the solution of the original mathematical question is obtained.

2.1 Internal Representation

We use an internal representation language, called Verb Connection Formula (VCF), to bridge the gap between mathematical questions and their formal mathematical counterparts.

VCF is based on assertional logic (Zhou, 2017), in which all mathematical objects are formalized as either individuals (constants and variables), or concepts, or operators (functions and relations). For instance, the natural language sentence "the integer x equals to 3" is transformed to "Integer(x), Equal(x,3)" in VCF, where "x" and "3" are individuals, "Integer" is a concept and "Equal" is a Boolean operator, i.e., relation. Meta level mathematical concepts such as equation and inequality are represented as concepts in VCF too. For instance, an equation $9 + 3^{n+2} = m$ in the question is transformed to "Equation(9+3**(n+2)=m)" in VCF, and an inequality $xyz \neq 0$ is transformed into "Inequality(x*y*z!=0)". VCF uses the symbol $:-$ for representing the implication relationship between statements. For instance, the VCF representation of the natural language sentence "When n is a positive integer, $9 + 3^{n+2} = m$" is "(Equation(9+3**(n+2)=m)):-(Positive(n),Integer(n))".

2.2 Translator

The translator transforms mathematical questions to corresponding statements in VCF.

Segmentation and POS tagging: Our translator uses the Stanford NLP parser (Chen and Manning, 2014) for segmentation and POS tagging. In order to handle Math questions, we introduce two new POS taggers, namely "FORM" for indicating Math formula and "VAL" for indicating variable. For example, the result of POS tagging of "When n is a positive integer, $9 + 3^{n+2} = m$" is "When/WRB n/VAL is/VBZ a/DT positive/JJ integer/NN ,/PUNCTUATION $9 + 3^{n+2} = m$/FORM".

Semantic parsing:

We adopt a top-down rule-based template approach for semantic parsing, i.e., translating math questions in English to statements in VCF. As shown in Table 1, we consider five basic clause types, corresponding to five syntactic structures respectively.

Type	Description	Examples
LEAF	a single word	"If","percent", "of"
NOUN	entity with adjunct words	"125 percent"
PREP	relation with multiple entities	125 percent of x
PRED	clause with operators	"125 percent of x is 150"
CONJ	causal connection between two clauses	"if 125 percent of x is 150,what is x percent of 75"

Table 1: Clause types

Algorithm 1 illustrates how to construct the semantic parsing tree for each sentence in the question from top to down. The root must be the sentence itself. Each node is assigned with one of the five types according to hand-crafted rule-based templates. Based on which, we decompose it into several child nodes correspondingly as different clause types result in different kinds of decomposition according to the templates. Note that each leaf node must be assigned with LEAF.

Algorithm 1: Semantic parsing algorithm

```
   input : a sentence in the question
   output: a list of VCF statements
1  root = original sentence;
2  Stack = Empty;
3  VCF_Stack = Empty;
4  Stack.push(root);
5  while Stack is not Empty do
6      current_node = Stack.pop();
7      templates = get_valid_templates(current_node);
8      template = choose_template(templates);
9      current_node.VCF_template = get_VCF_template(template);
10     VCF_Stack.push(current_node);
11     nodes = get_child_nodes(template,current_node);
12     for node in nodes do
13         current_node.child_nodes.add(node);
14         if type(node) != LEAF then
15             Stack.push(node);
16         end
17     end
18 end
19 while VCF_Stack is not Empty do
20     current_node = VCF_Stack.pop();
21     current_node.VCF=get_VCF_from_template_and_child_node();
22 end
23 return $root.VCF$
```

Figure 2 illustrates a semantic parsing tree example, in which each node is associated with one of the five basic types. According to this semantic parsing tree, we compute the resulting VCF statements from bottom to up recursively. For instance, the node "125 percent/NOUN" is converted into "NumberPer-

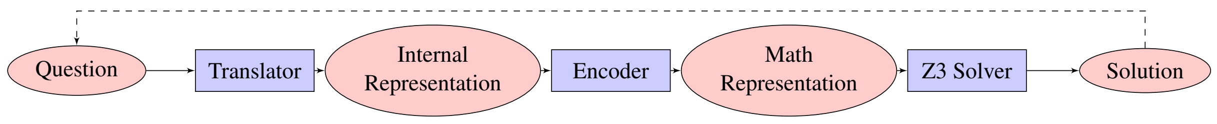


Figure 1: The architecture of `AiFu`

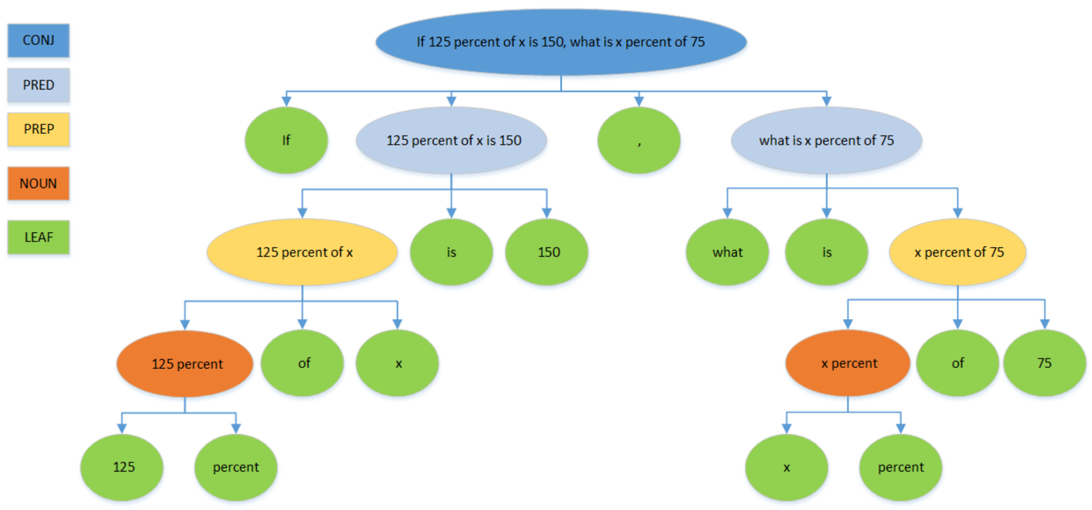


Figure 2: Semantic parsing tree: a case study

cent(125)", while the node "What is x percent of 75/PRED" is converted into "NumberPercent(x), Of(75,x,rs_a), Be(rs_b,rs_a), What(rs_b)". Finally, the root node, i.e., the original sentence, is converted into "(NumberPercent(x), Of(75,x,rs_a), Be(rs_b,rs_a), What(rs_b)) :- (NumberPercent(125), Of(x,125,rs_c), Be(rs_c,150))".

2.3 Encoder

The encoder further converts statements in VCF to formulas that can be accepted by Z3. Again, we use a rule based template approach for this purpose. There are two types of encoding templates. One is used to unify all different kinds of concepts/operators in VCF to a set of predefined concepts/operators that are accepted by Z3. For instance, a VCF statement "add(3,5,x)" is normalized as "Equal(3+5,x)". While the operator "add" is not a Z3 acceptable one, "Equal" and "+" are. Another type of template is to unify new entities that are created by VCF. For instance, the sentence "x is an integer" is converted to VCF statements "Be(rs_a,x),Integer(rs_a)" in the translator, which is further encoded as "Integer(x)" in the encoder by unifying the two entities "x" and "rs_a". At first glance, it seems tedious to introduce extra entities in the translator. However, this is exactly the reason why we need an intermediate representation language VCF because machines cannot directly understand what "x is an integer" really means.

By using the encoding templates, the VCF statements obtained in Figure 2 is converted into "Var: x:Real, Equations: x*125% = 150, Target: x%*75", which can be directly sent to the Z3 solver.

A large portion of SAT math questions can be done in this way, thus are suitable to use Z3 as the solver. Nevertheless, for some mathematical concepts such as progression, set, list and odd/even numbers, we need to make extra effort to formalize them in the modulo theory linear arithmetic. For instance, Odd(x) ("x is an odd number") can be converted into Equation($x\%2 = 1$). While the former cannot be directly encoded in linear arithmetic, the latter can.

2.4 Z3 Solver

Similar to some previous approaches (Hopkins et al., 2017), we also call the Z3 solver[1] as our reasoning engine. Z3 is a widely used Satisfiability Modulo Theories (SMT) solver, for solving problems that are represented in classical logic augmented with modulo theories, e.g., linear arithmetic.

However, Z3 has inherited difficulties on solving nonlinear equations, e.g., $3\sqrt{x} - 7 = 20$. In

[1]https://github.com/Z3Prover/z3

this case, we use SymPy [2] (Meurer et al., 2017), a Python library for symbolic mathematics, as the backup.

2.5 More on Geometry and Open Categories

For answering geometry questions, one has to understand diagrams. For this purpose, we first use the Optical Character Recognition (OCR) tool pytesseract[3] to obtain character information. Then, we follow the combined text and diagram understanding approach GeoS (Seo et al., 2015).

Understanding open-vocabulary algebra questions is a critical challenge. At first glance, it seems that the SAT open-vocabulary algebra subdataset is quite similar to Math23k (Wang et al., 2017). Hence, we attempted to use a Seq2Seq approach (Wang et al., 2017) that transforms math questions directly to their corresponding mathematical meanings. However, this attempt was not successful, mainly because of the following two reasons. First, questions in Math23k are much simpler. Second, Math23k is much larger in terms of volume.

Hence, we shifted back to a rule-based approach. We first use SVM to classify the type of questions. Based on which, regular expressions are used to transform questions in natural languages to statements in VCF, similar to that for the closed category described above. It turns out that its performance is slightly better than the Seq2Seq approach, yet still far from satisfactory.

2.6 Sub-symbolic System

We call the framework (see Figure 1) described above the "symbolic system" as it mainly uses a symbolic approach. However, some math questions remain unsolved. Hence, we also implement a guess system as a complementary counterpart. A simple guess system would be just a random guesser. Nevertheless, in `AiFu`, we use a neural-network based sub-symbolic approach, called the "sub-symbolic system" instead based on sentence embedding.

We treat the math question answering problem as a classification problem. We combine the question as well as a candidate choice into one sentence, and use a sentence embedding approach InferSent (Conneau et al., 2017) to embed it into a vector of 4096 dimensions. Then, we construct a simple four-layer fully-connected feed-forward neural network, in which the input is the 4096-dimension sentence embedding, the output is the 5 classes of answers and the two hidden layers both contain 1024 nodes. Finally, we train the network with the training dataset provided by the organizer.

3 Results

Table 2 reports the overall final results of `AiFu` on the test dataset. "Symb" is the symbolic system described from Sections 2.2 to 2.5, "Sub-S" is the sub-symbolic system described in Section 2.6, and "Integ" is the integrated system `AiFu` that combines them both by answering those questions not answered by the symbolic system with the sub-symbolic system. While "Acc" refers to the standard accuracy, "Pena Acc" refers to the penalized accuracy by deducting 0.25 points each for a wrong answer.

All in all, `AiFu` achieves an overall accuracy of 45% as well as an overall penalized accuracy of 36%. In particular, on the closed-vocabulary algebra category, it achieves a relatively high accuracy 70% (66% for penalized accuracy in contrast). The symbolic system plays a more critical role as it alone achieves 63% on answering closed-vocabulary algebra questions. More importantly, the symbolic system has a very high precision of 96%, thus has almost no penalization on all categories. In addition, unlike the sub-symbolic system, the symbolic system is fully explainable. However, it can be observed that all of these systems still perform poor on the Geometry and the open categories, especially when measured by penalized accuracy.

Table 3 illustrates the semantic parsing accuracy rate on closed-vocabulary algebra questions. It can be observed that, on the training and development datasets, we can achieve a relatively high accuracy of 84%. Nevertheless, it drops down to 71% on the test dataset. This is mainly because of the generalizability issue of templates.

In order to further analyze the genrealizability issue of templates, we consider the effects on the number of templates. Figure 3 shows how the template number affects the semantic parsing accuracy. It can be observed that, on the training dataset, the accuracy rates rapidly goes up to 50 % by the first 400 templates, then reaches 70% by 900 templates. Then, it slowly climbs up to 80% with 500 templates more. However, the growth

[2]https://www.sympy.org/en/index.html

[3]https://pypi.org/project/pytesseract/

Version	Acc (overall)	Acc (closed)	Acc (geo)	Acc (open)	Pena Acc (overall)	Pena Acc (closed)	Pena Acc (geo)	Pena Acc (open)
Symb	0.29	0.63	0.08	0.03	0.29	0.62	0.07	0.03
Sub-S	0.23	0.22	0.20	0.25	0.11	0.10	0.09	0.14
Integ	**0.45**	**0.70**	**0.26**	**0.26**	**0.36**	**0.66**	**0.14**	**0.16**

Table 2: Overall Results

Dataset	Closed
Training	84% (711/846)
Development	84% (187/222)
Test	71% (326/459)

Table 3: Semantic parsing accuracy

Error Analysis	Training
Non-linear question	19%
Contain math concept	49%
Contain definition	9%
Other tricky question	23%

Table 4: Questions that cannot be solved by Z3

rate drops down dramatically afterwards, making it very difficult to improve. The main reason is that SAT math questions have many long-tail questions that cannot be covered by ordinary templates.

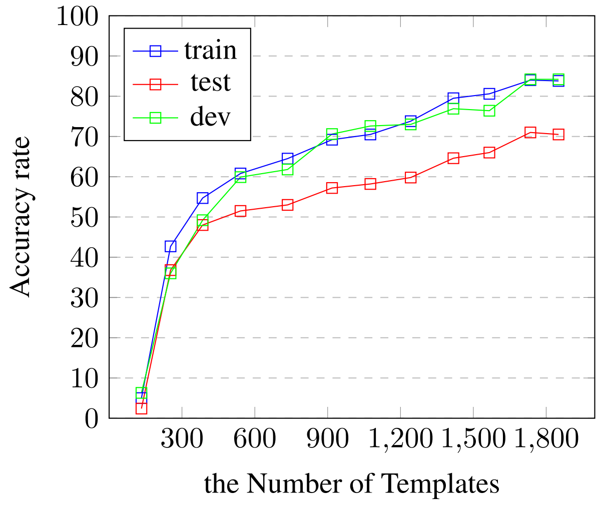


Figure 3: Effects on the number of templates

There are some questions that can be correctly parsed by our translator and encoder but failed to be solved by Z3. Among all, nearly 12% (100 out of 846) belong to this case in the training dataset. We further analyze their features. Among them, 49% need to use new math concepts, e.g., prime number, that cannot be represented in Z3. Around 19% of the questions are non-linear equations, which are very difficult for Z3. Finally, there are 9% of the questions need to define new objects, and the rest 23% are other kinds of tricky questions.

4 Conclusion

In this paper, we presents `AiFu`, a system that has won the first place in the SemEval-19 "Math Question Answering" competition. `AiFu` is a combined system that enhances a symbolic system with a sub-symbolic guesser. In `AiFu`, the symbolic system plays the most vital role, which itself can achieve a relatively high accuracy with many merits on closed-vocabulary algebra questions. Many state-of-the-art approaches are used and integrated in `AiFu`. Some new techniques are proposed including our new internal representation language VCF and our template structures.

For future work, the most important task is to improve the translator, which is the bottleneck, especially on geometry and open-vocabulary algebra questions. We believe that new foundations are needed, possibly requiring a deep integration of symbolic and sub-symbolic approaches.

References

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364.

Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An efficient smt solver. In *Proceedings* *of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'08/ETAPS'08, pages 337–340, Berlin, Heidelberg. Springer-Verlag.

Mark Hopkins, Ronan Le Bras, Cristian Petrescu-Prahova, Gabriel Stanovsky, Hannaneh Hajishirzi, and Rik Koncel-Kedziorski. Semeval-2019 task 10: Math question answering.

Mark Hopkins, Cristian Petrescu-Prahova, Roie Levin, Ronan Le Bras, Alvaro Herrasti, and Vidur Joshi. 2017. Beyond sentential semantic parsing: Tackling the math sat with a cascade of tree transducers. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 795–804. Association for Computational Linguistics.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 523–533.

Danqing Huang, Jing Liu, Chin-Yew Lin, and Jian Yin. 2018a. Neural math word problem solver with reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 213–223.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Danqing Huang, Jin-Ge Yao, Chin-Yew Lin, Qingyu Zhou, and Jian Yin. 2018b. Using intermediate representations to solve math word problems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 419–428.

Nate Kushman, Luke Zettlemoyer, Regina Barzilay, and Yoav Artzi. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 271–281.

Chao-Chun Liang, Yu-Shiang Wong, Yi-Chung Lin, and Keh-Yih Su. 2017. A goal-oriented meaning-based statistical multi-step math word problem solver with understanding, reasoning and explanation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 5235–5237.

Takuya Matsuzaki, Takumi Ito, Hidenao Iwane, Hirokazu Anai, and Noriko H. Arai. 2017. Semantic parsing of pre-university math problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 2131–2141.

Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. 2017. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103.

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Subhro Roy and Dan Roth. 2017. Unit dependency graph and its application to arithmetic word problem solving. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3082–3088.

Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving geometry problems: Combining text and diagram interpretation. In *EMNLP*.

Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. 2018. Mathdqn: Solving arithmetic word problems via deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5545–5552.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 817–822.

Yi Zhou. 2017. From first-order logic to assertional logic. In *Artificial General Intelligence - 10th International Conference, AGI 2017, Melbourne, VIC, Australia, August 15-18, 2017, Proceedings*, pages 87–97.

SemEval-2019 Task 12: Toponym Resolution in Scientific Papers

Davy Weissenbacher[†], **Arjun Magge**[‡], **Karen O'Connor**[†], **Matthew Scotch**[‡], **Graciela Gonzalez-Hernandez**[†]
[†]DBEI, The Perelman School of Medicine, University of Pennsylvania, Philadelphia, PA 19104, USA
[‡]Biodesign Center for Environmental Health Engineering, Arizona State University, Tempe, AZ 85281, USA
[†]{dweissen, karoc, gragon}@pennmedicine.upenn.edu
[‡]{amaggera, Matthew.Scotch}@asu.edu

Abstract

We present the SemEval-2019 Task 12 which focuses on toponym resolution in scientific articles. Given an article from PubMed, the task consists of detecting mentions of names of places, or toponyms, and mapping the mentions to their corresponding entries in GeoNames.org, a database of geospatial locations. We proposed three subtasks. In Subtask 1, we asked participants to detect all toponyms in an article. In Subtask 2, given toponym mentions as input, we asked participants to disambiguate them by linking them to entries in GeoNames. In Subtask 3, we asked participants to perform both the detection and the disambiguation steps for all toponyms. A total of 29 teams registered, and 8 teams submitted a system run. We summarize the corpus and the tools created for the challenge. They are freely available at https://competitions.codalab.org/competitions/19948. We also analyze the methods, the results and the errors made by the competing systems with a focus on toponym disambiguation.

1 Introduction

Toponym resolution, also known as geoparsing, geo-grounding or place name resolution, aims to assign geographic coordinates to all location names mentioned in documents. Toponym resolution is usually performed in two independent steps. First, toponym detection or geotagging, where the span of place names mentioned in a document is noted. Second, toponym disambiguation or geocoding, where each name found is mapped to latitude and longitude coordinates corresponding to the centroid of its physical location. Toponym detection has been extensively studied in named entity recognition: location names were one of the first classes of named entities to be detected in text (Piskorski and Yangarber, 2013). Disambiguation of toponyms is a more recent task (Leidner, 2007).

With the growth of the internet, the public adoption of smartphones equipped with Geographic Information Systems and the collaborative development of comprehensive maps and geographical databases, toponym resolution has seen an important gain of interest in the last two decades. Not only academic but also commercial and open source toponym resolvers are now available. However, their performance varies greatly when applied on corpora of different genres and domains (Gritta et al., 2018). Toponym disambiguation tackles ambiguities between different toponyms, like Manchester, NH, USA vs. Manchester, UK (Geo-Geo ambiguities), and between toponyms and other entities, such as names of people or daily life objects (Geo-NonGeo ambiguities). Additional linguistic challenges during the resolution step may be metonymic usage of toponyms, "91% of the US didn't vote for either Hilary or Trump" (a country does not vote, thus the toponym refers to the people living in the country), elliptical constructions, "Lakeview and Harrison streets" (the phrase refers to two street names Lakeview street and Harrison street), or when the context simply does not provide enough evidences for the resolution.

Although significant progress has been made in the last decade on toponym resolution, it is still difficult to determine precisely the current state-of-the-art performances (Leidner and Lieberman, 2011). As emphasized by several authors (Tobin et al., 2010; Speriosu, 2013; Weissenbacher et al., 2015; Gritta et al., 2018; Karimzadeh and MacEachren, 2019), the main obstacle is that few corpora of large size exist or are freely available. Consequently, researchers create their own (limited) corpora to evaluate their system, with the known drawbacks and biases that this implies.

Moreover, one corpus is not sufficient to evaluate a toponym resolver thoroughly, as the domain of a corpus strongly impacts the performance of a resolver. A disambiguation strategy can be optimal on one domain and damaging on another. In (Speriosu, 2013), Speriosu illustrates that toponyms occurring in historical literature will tend to resolve within a local vicinity, whereas toponyms occurring in international press news refer to the most prominent places by default. Otherwise additional information is provided to help the resolution (ex. Paris, the city in Texas).

In this article we first define the concept of toponym and detail the subtasks of this challenge (Section 3). Then, we summarize how we acquired and annotated our data (Section 4). In Section 5, after describing the evaluation metrics, we briefly describe the resources and the baseline system provided to the participants. In the last Section 6 we discuss the results obtained and the potential future direction for the task of toponym resolution.

2 Related Work

The *Entity Linking* task aims to map a name of an entity with the ID of the corresponding entity in a predefined Knowledge database (Bada, 2014). Entity linking has been largely studied by the community (Shen et al., 2015). Toponym resolution is a special case of the entity linking task where strategies dedicated to toponyms can improve overall performances. Three main strategies have been proposed in the literature. The first exploits the linguistic context where a toponym is mentioned in a document. The vicinity of the toponym often contains clues that help the readers to interpret it. These clues can be other toponyms (Tobin et al., 2010), other named entities (Roberts et al., 2010), or even more generally, specific topics associated more often with a particular toponym than with others (Speriosu, 2013; Adams and McKenzie, 2013; Ju et al., 2016). The second strategy relies on the physical properties of the toponyms to disambiguate their mentions in documents. The population heuristic or the minimum distance heuristic are popular heuristics using such properties. The population heuristics disambiguates toponyms by taking, among the ambiguous candidates, the candidate with the largest population, whereas the minimum distance heuristic disambiguates all toponyms in a document by taking the set of candidates that are the closest to each other (Leidner, 2007). A recent heuristic computes from Wikipedia a network expressing important toponyms and their semantic relation with other entities. The network is then used to disambiguate jointly all toponyms in a document (Hoffart and Weikum, 2013; Spitz et al., 2016). The last strategy is less frequently used as it depends on metadata describing the documents where toponyms are mentioned. These metadata are of various kinds, but they all indicate, directly or not, geographic areas to help interpret toponyms mentioned in documents. Such metadata can be geotagging of social media posts (Zhang and Gelernter, 2014) or external databases structuring the information detailed in a document (Weissenbacher et al., 2015). These three strategies are complementary and can be unified with machine learning algorithms as shown by (Santos et al., 2015) or (Kamalloo and Rafiei, 2018).

3 Task Description

The definition of toponym is still in debate among researchers. In its simpler definition, a toponym is a proper name of an existing populated place on Earth. This definition can be extended to include a place or geographical entity that is named, and can be designated by a geographical coordinate[1]. This encompasses cities and countries, but also lakes or monuments. In this challenge we consider the extended definition of toponyms and exclude all indirect mentions of places such as "30 km north from Boston", as well as metonymic usage and elliptical constructions of toponyms.

Subtask 1: Toponym Detection Toponym detection consists of detecting the text boundaries of all toponym mentions in full PubMed articles. For example, given the sentence `An H1N1 virus was isolated in 2009 from a child hospitalized in Nanjing, China.`, a perfect detector, regardless how, would return two pairs encoding the starting and ending positions of Nanjing and China, *i.e.* (64, 70) and (73, 77). Despite major progress, toponym detection is still an open problem and it was evaluated in a separate subtask since it determines the overall performance of the resolution. Toponym mentions missed during the detection cannot be disambiguated (False Negative, FN) and, inversely,

[1] https://unstats.un.org/unsd/geoinfo/UNGEGN/

phrases wrongly detected as toponyms will received geocoordinates during the disambiguation (False Positive, FP). Both FNs and FPs degrade the quality of the overall resolution.

Subtask 2: Toponym Disambiguation The second subtask focuses on the disambiguation of the toponyms only. In this subtask, all names of locations in articles are known by a disambiguator but not their precise coordinates. The disambiguator has to select the GeoNames IDs corresponding to the expected places among all possible candidates. GeoNames[2] is a crowdsourced database of geospatial locations and freely available. Following with our previous example, given the position of Nanjing in the sentence, a perfect disambiguator, regardless how, would have to choose among 12 populated places named Nanjing located in China in GeoNames and return the entry 7843770 in GeoNames. The disambiguator has to infer the expected place based on all information available in the article and not only based on the sentence. This subtask allows one to measure the performance of the disambiguation algorithms independently from the performances of the toponym detector used upstream.

Subtask 3: End-to-end, Toponym Resolution The last subtask evaluates the toponym resolver as it would be when deployed in real-world applications. Only the full PubMed articles are given to the resolver and all toponyms detected and disambiguated by the resolver are evaluated.

4 Data and Resources

4.1 A Case Study: Epidemiology of Viruses

The automatic resolution of the names of places mentioned in textual documents has multiple applications and, therefore, has been the focus of research for both industrial and academic organizations. For this challenge, we chose a scientific domain where the resolution of the names of places is key: epidemiology.

One aim in epidemiology might be to create maps of the locations of viruses and their migration paths, a tool which is used to monitor and intervene during disease epidemics. To create maps of viruses, researchers often use geospatial metadata of individual sequence records in public databases such as NIH's GenBank (Benson et al., 2017)[3]. The metadata provides the location of the infected host. With more than 3 million virus sequences[4], GenBank provides abundant information on viruses. However, previous work has suggested that geospatial metadata, when it is not simply missing, can be too imprecise for local-scale epidemiology (Scotch et al., 2011). In their article Scotch et al., 2011 estimate that only 20% of GenBank records of zoonotic viruses contain detailed geospatial metadata such as a county or a town name (zoonotic viruses are viruses able to infect naturally hosts of different species, like rabies). Most GenBank records provide generic information, such as Japan or Australia, without mentioning the specific places within these countries. However, more specific information about the locations of the viruses may be present in articles which describe the research work. To create a complete map, researchers are then forced to read these articles to locate in the text these additional pieces of geospatial metadata for a set of viruses of interest. This manual process can be highly time-consuming and labor-intensive.

This challenge was an opportunity to assess the development and evaluation of automated approaches to retrieve geospatial metadata with finer level of granularity from full-text journal articles, approaches that can be further transferred or adapted to resolve names of places in other scientific domains.

4.2 Corpus Collection

Our corpus is composed of 150 full text journal articles downloaded from the subset of PubMed Central (PMC) in open access[5]. All articles in this subset of PMC are covered by a Creative Commons license and free to access. We built our corpus using three queries on GenBank.

Subset A: For the first 60 articles, we downloaded 102,949 GenBank records that were linked to NCBI taxonomy id 197911 for influenza A. The downloaded records were associated with 1,424 distinct PubMed articles and 598 of them had links

[2] https://www.geonames.org/

[3] For this competition we chose to work with PubMed articles and the GenBank database as they provide more complete and detailed information for epidemiology than public health reports.

[4] Last accessed April 2019 with the query: https://www.ncbi.nlm.nih.gov/nuccore/?term=txid10239[Organism:exp]

[5] https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/

to an open access journal article in PubMed Central (PMC). We randomly sampled 60 articles from this set of 598 articles for manual annotation.

Subset B: We selected 60 additional articles by expanding our search to GenBank records linked to influenza B and C, rabies, hantavirus, western equine encephalitis, eastern equine encephalitis, St. Louis encephalitis, and West Nile virus. Our query returned a total of 544,422 GenBank records. We randomly selected a subset of records associated with 1,915 unique open access PMC articles. From these 1,915 articles, we randomly selected for toponym annotation a stratified sample of 60 articles, where strata were based on the number of GenBank records associated with the articles.

Subset C: We completed our corpus with 30 biomedical research articles to decrease bias and increase the generalizability of our corpus beyond toponym mentions in virus related research articles. From the 1,341 research articles returned by the search in PMC of the journal titles with the Article Attribute of Open access, we randomly selected 30 articles from top epidemiology journals, as determined by their impact factor in September 2018.

Since the 60 articles from Subset A had been used in our prior publications (Weissenbacher et al., 2015, 2017), we kept them all for training. We randomly selected half of the articles from Subset B and Subset C for training and left the second half for testing. The resulting corpus of 105 articles for training and 45 for testing was used for all three subtasks of the competition. The corpus is available for download on the Codalab used for the competition: `https://competitions.codalab.org/competitions/20171#learn_the_details-data_resources`.

4.3 Annotation Process

To perform the annotation, we manually downloaded the PDF versions of the PMC articles and converted them to text files using the freely available tool, Pdf-to-text[6]. We formatted the output to be compatible with the BRAT annotator[7] (Stenetorp et al., 2012). We manually detected and disambiguated the toponyms using GeoNames. We annotated toponyms in titles, bodies, tables and captions sections of the documents. We removed contents that would not contain virology-related toponyms, such as the names of the authors, acknowledgments and references, this was done manually. In cases where a toponym could not be found in GeoNames, we set its coordinates to a special value N/A. Prior to beginning annotation, we developed a set of annotation guidelines after discussion among three annotators. The resulting guidelines are also available in the Codalab of the competition. Two annotators were undergraduate students in biomedical informatics and biology, respectively, and our senior annotator has a M.S. in biomedical informatics.

Two annotators annotated independently 58 articles of Subset B to estimate the inter-annotator agreement. Since the detection task is a named-entity recognition task, we followed the recommendations of Rothschild and Hripcsak (2005) and used precision and recall metrics to estimate the inter-annotator rate. The inter-annotator agreement rate on the toponym detection was .94 precision (P) and .95 recall (R) which indicates a good agreement between the annotators. The inter-annotator agreement rate on the toponym disambiguation was 0.857 Accuracy. Subset C was also annotated by two annotators, although not independently, to ensure the quality of the annotation of all documents occurring in the test set of the competition.

The corpus contains a total of 1,506 distinct toponyms for a total of 8,360 occurrences. 1,228 of these toponyms occur in only one document (a document may include multiple occurrences). The average number of occurrences for a toponym is 5.5 with *China* being the most mentioned toponym with a total of 417 occurrences. The average ambiguity is about 26.3 candidates per toponym which is comparable to the average ambiguity found in existing corpora (Speriosu, 2013). The location *San Antonio* was the most ambiguous with 2633 possible candidates. 232 toponyms (531 occurrences) were not found in GeoNames using a strict match, this was caused by multiple reasons, like misspellings, non standard-abbreviations, missing entries in GeoNames, etc. 142 countries and continents are mentioned in our corpus with a total of 3,105 occurrences. The resolution of country and continent names are easier than other places but they represent only 37% of the total of the occurrences, making our corpus challenging.

[6]`http://www.foolabs.com/xpdf/download`
[7]`http://brat.nlplab.org/index.html`

5 Evaluation

5.1 Toponym Resolution Metrics

When a gold standard corpus and a toponym resolver are aligned on the same geographical database, here the database GeoNames, the standard metrics of precision, recall and F-measure can be used to measure the performance of the resolver. For this challenge, we report all results by using two common variations of these metrics: strict and overlapping measures. In the strict measure, resolver annotations are considered matching with the gold standard annotations if they hit the same spans of text; whereas in overlapping measure, both annotations match when they share a common span of text.

We computed the P and R for toponym detection with the standard equations: $Precision = TP/(TP+FP)$ and $Recall = TP/(TP+FN)$, where TP (True Positive) is the number of toponyms correctly identified by a toponym detector in the corpus, FP (False Positive) the number of phrases incorrectly identified as toponyms by the detector, and FN (False Negative) the number of toponyms not identified by the detector.

To evaluate the toponym disambiguation, we modified the equations computing the P and R used for toponym detection in order to account for both detection and disambiguation errors. The precision of the toponym disambiguation is given by the equation: $Pds = TCD/TCD + TID$, where TCD is the number of toponyms correctly identified and disambiguated by the toponym disambiguator in the corpus and TID is the number of toponyms incorrectly identified or incorrectly disambiguated in the corpus. The recall of the toponym disambiguation was computed by the equation: $Rds = TCD/TN$, where TN is the total number of toponyms in the corpus. F1ds is the harmonic mean of Pds and Rds. Since the resolvers competing and the gold corpus annotations were aligned on GeoNames, toponyms correctly identified were known by a simple match between the place IDs retrieved by the resolvers and those annotated by the annotators.

5.2 Baseline System

We released an end-to-end system to be used as a strong baseline. This system performs sequentially the detection and the disambiguation of the toponyms in raw texts. To detect the toponyms the system uses a feedforward neural network described in (Magge et al., 2018). The disambiguation of all toponyms detected is then performed using a common heuristic, the population heuristic. Using this heuristic, the system always disambiguates a toponym by choosing the place which has the highest population in GeoNames. The baseline system can be downloaded from the Codalab website of the competition. We also made available to the participants a Rest service to search a recent copy of GeoNames, the documentation and the code to deploy the service locally can be found on the Codalab website.

6 Systems

6.1 Results

Twenty nine teams registered to participate in the shared-task and eight teams submitted. 21/8/13 submissions from 8/4/6 teams were included in the final evaluations of sub-task 1/2/3 respectively. All systems which attempted to resolve the toponyms in Subtask 3 opted for a pipeline architecture where the detection and the disambiguation steps were performed independently and sequentially. Table 1 summarizes the characteristics of the systems along with their use of external resources. Tables 2, 3 and 4 presents the performances for each team. Team DM_NLP achieved the best performances on all sub-tasks (Wang et al., 2019).

Toponym Detection: With all systems but one, Deep Recurrent Neural Networks were the most commonly used and efficient technology to detect toponyms in our corpus. Their architectures varied with respect to the integration of character embedding layers, mechanisms of attention, integration of external features (such as POS tagging or other Named Entities) or the choice of a general or in domain corpus for pre-training their word and sentence embeddings. In our epidemiological corpus, toponyms were not only mentioned in the body of the articles but also in tables. And interestingly, top ranked systems detected the toponyms with two different algorithms, one dedicated to the body and one to the tables of the articles. The top ranking system outperformed other competitors for Subtask 1 significantly, with a margin of 4 points separating it from the second ranked system, even though the same technology was used. Both teams used dedicated algorithms for bodies and tables but Team DM_NLP implemented several strategies to improve the pre-training of their

Toponym Detection		
Rank	**Team**	**System details**
1	DM_NLP	*Architecture:* Ensemble of C_biLSTM + W_biLSTM + FF + CRF
	(Alibaba Group)	*Details:* word2vec/ELMo embeddings, POS + NE + Chunk features
		Resources: OntoNote5.0, CoNLL'13 and Weakly labeled training corpora
3	UniMelb	*Architecture:* W_biLSTM + FF + SoftMax
	(University of Melbourne)	*Details:* Glove/ELMo embeddings, Self-Attention
		Resources: WikiNER, inhouse gazetteer of place name abbreviations
		& organization names classifier
4	UArizona	*Architecture:* C_biLSTM + W_biLSTM + CRF
	(University of Arizona)	*Details:* Glove embeddings, affixes features
		Resources: Weakly labeled training corpus
5	THU_NGN	*Architecture:* Ensemble of C_CNN + W_biLSTM + CRF
	(Tsinghua University)	*Details:* Glove/Word2Vect/FastText/ELMo/Bert embeddings,
		LM + POS + Lexicon features
6	UNH	*Architecture:* 1. W_biLSTM + CRF; 2. W_CNN + FF + sigmoid;
	(University of New Hampshire)	3. W_FF + sigmoid
		Details: word2vec/ELMo embeddings, orthographic + lexicon features
8	RGCL-WLV	*Architecture:*1. W_biGRU + capsule + FF + sigmoid;
	(University of Wolverhampton/	2. W_biLSTM + W_biGRU + FF + sigmoid; 3. traditional classifiers
	Universidad Politecnica de Madrid)	*Details:* word2vec embeddings, Self-attention
		Resources: ANNIE's gazetteer of regions
		& inhouse gazetteer of US regions and abbreviations
Toponym Disambiguation		
Rank	**Team**	**System details**
1	DM_NLP	*Strategy:* Ranking candidates + Stacking LightGBM classifiers
		External Resources: Wikipedia
3	UniMelb	*Strategy:* Ranking candidates + SVM Classifier
4	UArizona	*Strategy:* Population heuristic
6	THU_NGN	*Strategy:* Toponym frequencies + population heuristic

Table 1: System and resource descriptions for toponym resolution[8].

[8] We use C_biLSMT and C_CNN to denote bidirectonal LSTMs or CNNs encoding sequences of characters, W_biLSTM, W_biGRU and W_FF to denote bidirectional LSTMs/GRUs or Feed Forward encoders of word embeddings.

	Strict macro			**Strict micro**			**Overlap macro**			**Overlap micro**		
Team	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
DM_NLP	**.9265**	**.9060**	**.9161**	**.9292**	**.8564**	**.8913**	**.9456**	**.9238**	**.9346**	**.9539**	**.8797**	**.9153**
DM_NLP	.9214	.9010	.9111	.9222	.8512	.8853	.9447	.9224	.9334	.9510	.8776	.9128
DM_NLP	.9201	.9000	.9100	.9117	.8479	.8786	.9419	.9204	.9311	.9412	.8756	.9072
UniMelb	.8827	.8598	.8711	.8469	.7748	.8092	.9222	.8911	.9064	.9135	.8283	.8688
QWERTY	.9015	.8426	.8710	.8935	.7808	.8333	.9277	.8622	.8937	.9258	.8096	.8638
UArizona	.8869	.8073	.8452	.8797	.7068	.7838	.9084	.8268	.8657	.9114	.7357	.8142
UArizona	.8803	.8079	.8426	.8792	.7131	.7875	.9027	.8279	.8637	.9099	.7412	.8169
UArizona	.8897	.7960	.8403	.8825	.6972	.7790	.9112	.8152	.8606	.9144	.7262	.8095
THU_NGN	.8897	.7818	.8323	.8647	.6615	.7496	.9221	.8125	.8639	.9136	.7025	.7943
THU_NGN	.8951	.7743	.8303	.8745	.6489	.7450	.9257	.8015	.8592	.9186	.6849	.7847
THU_NGN	.8966	.7699	.8284	.8715	.6497	.7444	.9254	.7961	.8559	.9197	.6892	.7879
UNH	.8616	.7810	.8193	.8354	.6500	.7312	.9100	.8189	.8620	.8968	.7035	.7885
UniMelb	.8402	.7967	.8179	.8023	.6768	.7342	.8866	.8398	.8626	.8795	.7440	.8061
UNH	.8360	.7374	.7836	.8073	.6175	.6998	.9079	.7882	.8438	.9132	.6971	.7906
Baseline	.8246	.7345	.7770	.8032	.5973	.6851	.8989	.7810	.8358	.9038	.6719	.7708
UNH	.8111	.7403	.7741	.7819	.6459	.7074	.8859	.7984	.8399	.8904	.7372	.8066
NLP_IECAS	.8111	.6944	.7482	.7807	.5414	.6394	.8601	.7187	.7831	.8421	.5808	.6874
NLP_IECAS	.7527	.7226	.7373	.7298	.5796	.6461	.8209	.7700	.7946	.8155	.6457	.7207
NLP_IECAS	.7395	.7334	.7364	.7270	.5853	.6485	.8101	.7824	.7960	.8143	.6553	.7262
RGCL-WLV	.8392	.4911	.6196	.8210	.3505	.4913	.9032	.5117	.6533	.8926	.3743	.5274
RGCL-WLV	.8200	.4844	.6090	.8021	.3464	.4839	.8928	.5082	.6477	.8850	.3746	.5264
RGCL-WLV	.8280	.4746	.6034	.8168	.3396	.4798	.8980	.4969	.6398	.8936	.3654	.5187

Table 2: Results of the toponym detection task, Subtask 1.

system which, according to their ablation study (Wang et al., 2019), proved to be effective[9]. Note that the performance of the first system is close to our IAA for toponym detection.

Toponym Disambiguation: All systems relied on handcrafted features to disambiguate toponyms. Their features described the lexical context of the toponyms and their importance. The importance of the toponyms was estimated by the frequencies of the candidates in the training data or by their populations. While the two top ranked systems combined such features with machine learning, SVM for UniMelb and a gradient boosting algorithm for DM_NLP, others just encoded them into hard rules leading to suboptimal disambiguation.

6.2 Analysis

We analyzed a sample of errors to understand the remaining challenges for toponym disambiguation systems based on the results of Sub-task 2. We randomly selected 10 articles and analyzed 103 mentions of toponyms disambiguated incorrectly by all systems. We manually found 5 distinct categories of errors. For the largest category of errors, with 62 cases, the systems missed context clues used by the authors of the articles to convey the correct interpretation of the toponym and chose the wrong candidates. Such clues include the mention of a country in the header of a table or the explicit mention of a district after an ambiguous toponym. 17 errors were due to the systems not complying with the guidelines, selecting instead populated places or cities when the expected choices were toponyms with a higher administrative level. 8 candidates were not found in GeoNames by strict or fuzzy matching because of their surface forms. These were unconventional abbreviations, rare acronyms or words split by a hyphen. Despite our efforts to limit annotation errors, 15 were found in our sample[10]. The last error was a toponym where the choice made by the annotators can be argued.

[9]Team QWERTY did not describe their system at the time of writing. We were therefore unable to compare it with other systems.

[10]Since we analyzed entire articles, this count includes multiple mentions of the same toponym repeatedly annotated with the same error

7 Conclusion

In this paper we presented an overview of the results of SemEval 2019 Task 12 which focuses on toponym resolution in scientific articles. Given an article from PubMed, the task consists of detecting all mentions of place names, or toponyms, in the article and mapping them to their corresponding entry in GeoNames, a database of geospatial locations. All systems resolved the toponyms in our corpus sequentially, detecting the toponyms before disambiguating them. Among the 21 systems submitted for toponym detection, neural network based approaches were the most popular and the most efficient to detect toponyms with scores approaching the Inter-Annotator agreement. One key to success for the top ranked systems was to design two different algorithms to detect toponyms in the body and in the tables of the articles. The disambiguation of the toponyms remains challenging. Despite a clever use of rules or machine learning to combine features describing the lexical context of the toponyms and their importance from the 4 competing systems, the strict macro F1ds score of .82 of the best system signals space for improvement. Our analysis of common disambiguation errors reveals that it is still difficult for the systems to capture linguistic evidence in the context of the toponyms that dictate their disambiguation, causing 60% of the errors of the systems. The end-to-end performance of the best toponym resolver was .77 F1ds strict macro, a score high enough for scientists to benefit from automation to reduce their workload when extracting toponyms from the voluminous and quickly growing literature, while still leaving room for technical improvement.

Funding

Research reported in this publication was supported by the National Institute of Allergy and Infectious Diseases of the National Institutes of Health under Award Number R01AI117011 to M.S. and G.G. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

	Strict macro	Strict micro
Team	**F1ds**	**F1ds**
DM_NLP	**.8234**	.7781
DM_NLP	.8215	**.7821**
UniMelb	.8180	.7759
UniMelb	.8180	.7759
DM_NLP	.8070	.7521
Baseline	.7400	.6768
NLP_IECAS	.7233	.6582
NLP_IECAS	.7230	.6607
THU_NGN	.6721	.5886

Table 3: Results of the toponym disambiguation task, Subtask 2.

Toponym Detection												
	Strict macro			**Strict micro**			**Overlap macro**			**Overlap micro**		
Team	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
DM_NLP (run 2)	**.9265**	.9060	**.9161**	**.9292**	.8564	.8913	**.9456**	.9238	.9346	**.9539**	.8797	.9153
QWERTY (run 1)	.9203	**.9095**	.9148	.9214	**.8706**	**.8953**	.9438	**.9311**	**.9374**	.9501	**.8972**	**.9229**
DM_NLP (run 1)	.9214	.9010	.9111	.9222	.8512	.8853	.9447	.9224	.9334	.9510	.8776	.9128
DM_NLP (run 3)	.9201	.9000	.9100	.9117	.8479	.8786	.9419	.9204	.9311	.9412	.8756	.9072
UniMelb (run 2)	.8821	.8598	.8708	.8464	.7748	.8090	.9215	.8911	.9061	.9130	.8283	.8686
UniMelb (run 1)	.8884	.8124	.8487	.8767	.6986	.7776	.9349	.8442	.8872	.9322	.7448	.8280
UArizona (run 3)	.8869	.8073	.8452	.8797	.7068	.7838	.9084	.8268	.8657	.9114	.7357	.8140
UArizona (run 2)	.8803	.8079	.8426	.8792	.7131	.7875	.9027	.8279	.8637	.9099	.7412	.8169
UArizona (run 1)	.8897	.7960	.8403	.8825	.6972	.7790	.9112	.8152	.8606	.9144	.7262	.8095
THU_NGN (run 1)	.8951	.7743	.8303	.8745	.6489	.7450	.9257	.8015	.8592	.9186	.6849	.7847
Baseline	.8246	.7345	.7770	.8032	.5973	.6851	.8989	.7810	.8358	.9038	.6719	.7708
NLP_IECAS (run 2)	.8111	.6944	.7482	.7807	.5414	.6394	.8601	.7187	.7831	.8421	.5808	.6874
NLP_IECAS (run 3)	.8111	.6944	.7482	.7807	.5414	.6394	.8601	.7187	.7831	.8421	.5808	.6874
NLP_IECAS (run 1)	.7527	.7226	.7373	.7298	.5796	.6461	.8209	.7700	.7946	.8155	.6457	.7207
Toponym Disambiguation												
	Strict macro			**Strict micro**			**Overlap macro**			**Overlap micro**		
Team	**Pds**	**Rds**	**F1ds**	**Pds**	**Rds**	**F1ds**	**Pds**	**Rds**	**F1ds**	**Pds**	**Rds**	**F1ds**
DM_NLP (run 2)	**.7840**	**.7661**	**.7749**	**.7601**	**.7005**	**.7291**	**.7887**	**.7715**	**.7800**	**.7646**	**.7060**	**.7341**
DM_NLP (run 1)	.7762	.7587	.7674	.7513	.6934	.7212	.7840	.7667	.7753	.7593	.7019	.7295
QWERTY (run 1)	.7597	.7506	.7551	.7336	.6931	.7128	.7677	.7586	.7631	.7417	.7016	.7211
UniMelb (run 2)	.7437	.7276	.7355	.6848	.6268	.6545	.7510	.7368	.7438	.6964	.6399	.6670
UniMelb (run 1)	.7286	.6711	.6987	.6876	.5479	.6098	.7331	.6777	.7043	.6941	.5564	.6177
UArizona (run 3)	.6773	.6225	.6487	.6514	.5233	.5804	.6761	.6242	.6491	.6507	.5253	.5813
UArizona (run 2)	.6739	.6243	.6482	.6533	.5299	.5852	.6725	.6256	.6482	.6521	.5313	.5855
UArizona (run 1)	.6823	.6149	.6468	.6600	.5214	.5826	.6807	.6164	.6470	.6586	.5231	.5831
Baseline	.6605	.5912	.6240	.6252	.4649	.5333	.6787	.6071	.6409	.6505	.4857	.5561
THU_NGN (run 1)	.6581	.5738	.6131	.6052	.4491	.5156	.6605	.5784	.6167	.6070	.4537	.5193
NLP_IECAS (run 2)	.6527	.5584	.6019	.6339	.4395	.5191	.6631	.5666	.6111	.6504	.4510	.5326
NLP_IECAS (run 3)	.6529	.5582	.6018	.6378	.4423	.5223	.6633	.5664	.6110	.6543	.4537	.5359
NLP_IECAS (run 1)	.5852	.5626	.5737	.5634	.4474	.4988	.5935	.5717	.5824	.5772	.4603	.5120
DM_NLP (run 3)	.0279	.0278	.0279	.0305	.0284	.0294	.0314	.0312	.0313	.0354	.0330	.0342

Table 4: Results of the toponym resolution task, Subtask 3.

References

Benjamin Adams and Grant McKenzie. 2013. *Inferring Thematic Places from Spatially Referenced Natural Language Descriptions*. Springer Netherlands.

Michael Bada. 2014. Mapping of biomedical text to concepts of lexicons, terminologies, and ontologies. *Methods in Molecular Biology: Biomedical Literature Mining*, 1159:33–45.

Dennis A. Benson, Mark Cavanaugh, Karen Clark, Ilene Karsch-Mizrachi, David J. Lipman, James Ostell, and Eric W. Sayers. 2017. Genbank. *Nucleic Acids Research*, 45(D):37–42.

Milan Gritta, Mohammad T. Pilehvar, Nut Limsopatham, and Nigel Collier. 2018. What's missing in geographical parsing? *Language Resources and Evaluation*, 52(2):603–623.

Johannes Hoffart and Gerhard Weikum. 2013. Discovering and disambiguating named entities in text. In *Proceedings of the 2013 SIGMOD/PODS Ph.D. Symposium*, SIGMOD'13 PhD Symposium, pages 43–48. ACM.

Yiting Ju, Benjamin Adams, Krzysztof Janowicz, Yingjie Hu, Bo Yan, and Grant Mckenzie. 2016. Things and strings: Improving place name disambiguation from short texts by combining entity co-occurrence with topic modeling. In *20th International Conference on Knowledge Engineering and Knowledge Management - Volume 10024*, EKAW 2016, pages 353–367. Springer-Verlag New York, Inc.

Ehsan Kamalloo and Davood Rafiei. 2018. A coherent unsupervised model for toponym resolution. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pages 1287–1296. International World Wide Web Conferences Steering Committee.

Morteza Karimzadeh and Alan M. MacEachren. 2019. Geoannotator: A collaborative semi-automatic platform for constructing geo-annotated text corpora. *ISPRS International Journal of Geo-Information*, 8(4).

Jochen L. Leidner. 2007. *Toponym Resolution in Text: Annotation, Evaluation and Applications of Spatial Grounding of Place Names*. Ph.D. thesis, Institute for Communicating and Collaborative Systems School of Informatics, University of Edinburgh.

Jochen L. Leidner and Michael D. Lieberman. 2011. Detecting geographical references in the form of place names and associated spatial natural language. *SIGSPATIAL*, 3(2):5–11.

Arjun Magge, Davy Weissenbacher, Abeed Sarker, Matthew Scotch, and Graciela Gonzalez-Hernandez. 2018. Deep neural networks and distant supervision for geographic location mention extraction. *Bioinformatics*, 34(13):i565–i573.

Jakub Piskorski and Roman Yangarber. 2013. Information extraction: Past, present and future. In Thierry Poibeau, Horacio Saggion, Jakub Piskorski, and Roman Yangarber, editors, *Multi-source, Multilingual Information Extraction and Summarization*, Theory and Applications of Natural Language Processing, pages 23–49. Springer Berlin Heidelberg.

Kirk E. Roberts, Cosmin A. Bejan, and Sanda M. Harabagiu. 2010. Toponym disambiguation using events. In *FLAIRS Conference*.

Adam S. Rothschild and George Hripcsak. 2005. Agreement, the f-measure, and reliability in information retrieval. *Journal of the American Medical Informatics Association*, 12(3):296–298.

João Santos, Ivo Anastácio, and Bruno Martins. 2015. Using machine learning methods for disambiguating place references in textual documents. *GeoJournal*, 80(3):375–392.

Matthew Scotch, Indra N. Sarkar, Changjiang Mei, Robert Leaman, Kei-Hoi Cheung, Pierina Ortiz, Ashutosh Singraur, and Graciela Gonzalez. 2011. Enhancing phylogeography by improving geographical information from genbank. *Journal of Biomedical Informatics*, 44(44-47).

Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transaction on Knowledge and Data Engineering*, 27(2).

Michael A. Speriosu. 2013. *Methods and Applications of Text-Driven Toponym Resolution with Indirect Supervision*. Ph.D. thesis, University of Texas.

Andreas Spitz, Johanna Geiß, and Michael Gertz. 2016. So far away and yet so close: Augmenting toponym disambiguation and similarity with text-based networks. In *Proceedings of the Third International ACM SIGMOD Workshop on Managing and Mining Enriched Geo-Spatial Data*, GeoRich '16, pages 2:1–2:6. ACM.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: A web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL'12, pages 102–107. Association for Computational Linguistics.

Richard Tobin, Claire Grover, Kate Byrne, James Reid, and Jo Walsh. 2010. Evaluation of georeferencing. In *Proceedings of the 6th Workshop on Geographic Information Retrieval*, GIR '10, pages 7:1–7:8.

Xiaobin Wang, Chunping Ma, Huafei Zheng, Chu Liu, Pengjun Xie, Linlin Li, and Luo Si. 2019. Dm_nlp at semeval-2018 task 12: A pipeline system for toponym resolution. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Davy Weissenbacher, Abeed Sarker, Tasnia Tahsin, Matthew Scotch, and Graciela Gonzalez. 2017. Extracting geographic locations from the literature for virus phylogeography using supervised and distant supervision methods. In *In Proceedings of AMIA Joint Summits on Translational Science*.

Davy Weissenbacher, Tasnia Tahsin, Rachel Beard, Mari Figaro, Robert Rivera, Matthew Scotch, and Graciela Gonzalez. 2015. Knowledge-driven geospatial location resolution for phylogeographic models of virus migration. *Bioinformatics*, 31(12):i348–i356.

Wei Zhang and Judith Gelernter. 2014. Geocoding location expressions in twitter messages: A preference learning method. *J. Spatial Information Science*, 9:37–70.

Abbreviations

POS: Part-Of-Speech
NER: Named Entity Recognition
LM: Language Model
ANNIE: A Nearly-New Information Extraction
SVM: Support Vector Machine
CRF: Conditional Random Field
FF: Feedforward
CNN: Convolutional Neural Network
biLSTM: bidirectional Long Short-Term Memory
biGRU: bidirectional Gated Recurrent Unit

DM_NLP at SemEval-2019 Task 12: A Pipeline System for Toponym Resolution

Xiaobin Wang, Chunping Ma, Huafei Zheng, Chu Liu, Pengjun Xie, Linlin Li, Luo Si
Alibaba Group, China
{xuanjie.wxb, chunping.mcp, huafei.zhf, chuci.lc, chengchen.xpj, linyan.lll, luo.si}@alibaba-inc.com

Abstract

This paper describes DM-NLP's system for toponym resolution task at Semeval 2019. Our system was developed for toponym detection, disambiguation and end-to-end resolution which is a pipeline of the former two. For toponym detection, we utilized the state-of-the-art sequence labeling model, namely, BiLSTM-CRF model as backbone. A lot of strategies are adopted for further improvement, such as pre-training, model ensemble, model averaging and data augment. For toponym disambiguation, we adopted the widely used searching and ranking framework. For ranking, we proposed several effective features for measuring the consistency between the detected toponym and toponyms in GeoNames. Eventually, our system achieved the best performance among all the submitted results in each sub task.

1 Introduction

The toponym resolution task is aimed to detect toponyms in scientific papers and link them to entities in a geographical knowledge base (GeoNames[1] in this task). A toponym is a proper name of a place or geographical entity that is named, and can be designated by a geographical coordinate, including cities, countries, lakes or monuments.

We developed an end-to-end toponym resolution system (for subtask 3) which is a pipeline of toponym detection (for subtask 1) and disambiguation (for subtask 2). We model the detection task as a Named Entity Recognition (NER) and address it with popular sequence labeling framework. For disambiguation task, we adopted the searching and ranking framework which is widely used in Entity linking task.

Toponym is a special type of entity similar to the location entity in the general NER task. Thus, the well-studied NER models may be effective for detecting toponyms. The most successful NER models (Chen et al., 2006; Lample et al., 2016; Huang et al., 2015; Yao and Huang, 2016) are sequence labeling models, including the traditional CRF (Conditional Random Field (Lafferty et al., 2001)) and some variants of RNNs (Recurent Neural Networks) proposed recently, like LSTM-CRF, BiLSTM-CRF, BiLSTM-CNN-CRF, etc. In this paper, We utilize the most popular model, i.e., BiLSTM-CRF for toponym detection. Beyond the model, a prevalent pre-training embedding named ELMo is used after fine-tuning. Model averaging and model ensemble are used for avoiding overfitting. Data sets from other NER tasks are exploited to augment the training data. We also proposed a dictionary based method for detecting toponyms in tables separately. Since tables have some peculiarities, i.e., well formatted yet without meaningful context for toponyms in them.

Toponym disambiguation can be seen as a variant of entity linking (EL) problem, which links entity mentions in articles to entities in knowledge base (KB) like Wikipedia. A typical EL system consists of candidate entity generation and ranking as well as unlinkable mention prediction (Shen et al., 2015). The major challenge is that the KB of toponym lacks of background information other than toponym names, types and coordinates. Therefore, we follow the typical EL method (Hoffart et al., 2011) for toponym disambiguation and propose a classification based ranking method. Specifically, We recast the problem as a binary classification task asking that whether a toponym in GeoNames is a link for given toponym. If more than one positives exist, they are ranked according to their confidence scores. Coupled with the classifier, We introduce many features which measure the consistency between toponyms effectively, including name string similarity, candidate

[1] https://geonames.org

attributes, contextual features and mention list features.

Our contributions to this task can be summarized as follows:

- Proposing an approach to process tables separately from the main body.
- Proposing a novel data augment approach to exploit external data.
- Designing many novel and effective features for disambiguation.

2 Methodology

2.1 Overview

Our system for toponym resolution consists of toponym detection and disambiguation. The former is based on a sequence labeling model and is enhanced with pre-training, model ensemble and data augment. The later is a two-stage approach which obtains candidates by searching and does disambiguation via classification.

2.2 Toponym Detection

A scientific article usually contains a main body and tables. Detecting toponyms in these two types of content are different due to toponyms in tables lack of contextual information. Consequently, we adopt two different approaches.

2.2.1 Detection in Main Body

We recast the problem the Toponym Detection in main body as a Named Entity Recognition task and we make use of the BiLSTM-CRF model with the contextual information as input. To alleviate over-fitting, we apply model averaging training strategies. Finally, a voting method is utilized to benefit from multiple models.

Input Information Based upon our previous work (Ma et al., 2018) on sequence labeling, our system incorporates four types of linguistic information: Part-of-Speech (POS) tags, NER labels, Chunking labels and ELMo (Peters et al., 2018). The former three are generated by open source tools. In detail, we use Stanford CoreNLP (Manning et al., 2014) to annotate POS tags and NER labels, and use OpenNLP [2] to annotate Chunking labels. These information are represented as distributional vectors which are randomly initialized and trained with the entire model. ELMo

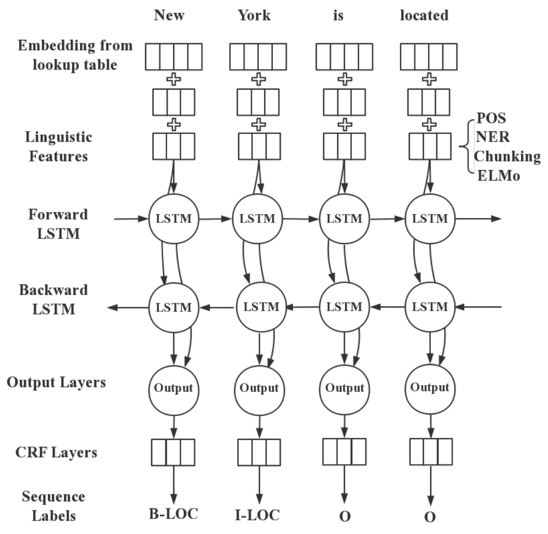


Figure 1: Architecture of BiLSTM-CRF model

is a deep contextualized word representation that models both complex characteristics of word use, and how these uses vary across linguistic contexts. These word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large corpus of texts. We fine tuned ELMo on the weakly labeled data provided by the organizers, so that the vectors will be adapted to this domain.

BiLSTM-CRF Model As illustrated in Figure 1, the entire model consists of five layers: word representation layer, input layer, feature extraction layer, output layer and CRF layer. The word representation layer is a group of BiLSTM with shared parameters. Each BiLSTM corresponds to a word. The BiLSTM takes a sequence of character (characters in a word) embedding as input and concatenates the final hidden states (forward and backward) as the representation of the word. Designing a neural network architecture with character representation as input is appealing for several reasons. Firstly, words which have the same morphological properties (like the prefix or suffix of a word) often share the same grammatical function or meaning. Secondly, a character-level analysis can help to address the out-of-vocabulary problem, Thirdly, capitalization may provide additional information. A recent study (Lample et al., 2016) has shown that BiLSTM is an effective approach to extract morphological information from characters of words, and consequently help to improve the performance

[2]`https://opennlp.apache.org/`

in NER and POS tagging.

The input layer generates the final representation of each word by concatenating three types of vectors, the pre-trained word embedding, the word vector given by the character BiLSTM and the vector of linguistic information (POS label, NE label, chunking label and ELMo vector).

The feature extraction layer is another BiLSTM. RNNs are well-studied solutions for a neural network to process variable length input and have a long term memory. As a variant of RNNs, the long-short term memory (LSTM) unit with three multiplicative gates allows highly non-trivial long-distance dependencies to be easily learned. Therefore, we use a bidirectional LSTM network as proposed in (Graves et al., 2013) to efficiently make use of past features (via forward states) and future features (via backward states) for a specific time frame.

The output layer is a fully connected feed forward network which outputs the probability distribution over all labels.

The CRF Layer is use on the top to decode the appropriate label sequence. For sequence labeling tasks, such as POS tagging or NER, the adjacent labels are often strongly related (e.g. I-ORG cannot follow B-PER or I-LOC in NER tasks like CoNLL2003). CRF model is good at modeling these constraints.

Model Averaging Random initialization and shuffling order of training sentences introduce randomization when training a model. During our experiments, we found that model predictions vary considerably even when the same pre-trained data and parameters are used. In order to utilize the power of model ensemble and avoid overfitting problem, we use a script provided by tensor2tensor to average values of variables in a list of checkpoint files generated by BiLSTM-CRF networks.

Ensemble By using different pre-trained word embeddings or using different linguistic information, we trained multiple models, we apply an average voting strategy to compute the final decision of our system from all models. Experimental result shows that voting indeed boosts the overall performance.

2.2.2 Detection in Tables

As important components of a scientific article, tables have specific formats:

- They usually begin with the word 'Table'.
- The first line is called the header which indicates the meaning of each column.
- All rows follow the schema defined by the header of the tables.

According to our analysis of the training data, many toponyms are mentioned in tables. Nevertheless, the contexts of these toponyms differ significantly from contexts of toponyms in main body. The later are always meaningful sentences. As a result, performances may drop significantly if a model trained to recognize toponyms in the sentences of the main body is used to recognized toponyms occuring in tables. Thus, we propose a novel approach for detecting toponyms in tables which are processed separately with details as follows:

1. Analyze the mean and variance of words counts (split by space), within a window of text. Decreasing the size of window until the variance is smaller than a threshold.
2. If the word 'Table' is found in the context of the window, take the n-gram within this window as toponym if it exists in GeoNames database.

2.2.3 Postprocess

Rule based postprocessing is applied in the end of the detection step to avoiding errors which occur frequently in development set. The following rules are applied to a toponym for generating possible corrections, which are confirmed and used to replace the original mention by figuring out whether a correction exists in GeoNames.

- If a word of locality, such as eastern, appears before a toponym within three words, we correct the candidate predicted by adding the word of locality to the toponym.
- If a toponym ends with a suffix word (e.g., Province) which indicates an administrative division, we make a candidate correction by removing the suffix when the suffix occurs in a predefined black list.
- If an abbreviation appears after a toponym and the abbriviation consists of of all the capital letters of the words composing the name of the toponym, we include the abbreviation as a new candidate toponym.

2.3 Toponym Disambiguation

Our approach for disambiguation has two stages. First, we retrieve possible candidate toponyms from GeoNames database using a search engine with a toponym mention as query. Second, a binary classifier with carefully designed features are applied to each candidate to figure out whether it is the appropriate place that the mention refers to.

2.3.1 Candidate Generation

This stage is based on an offline search engine implemented with Lucene[3]. All GeoNames records were indexed in advance. Then, we search the index with the toponym mentions given by the detection module as queries. In order to ensure higher recall rate, we addressed the alias issue. We expand the query by alternate names and enable fuzzy matching searching.

Alternative names of given toponym mentions are obtained by the following ways:

1. Alternative names recorded in GeoNames dump files, including allCountries, alternatenames, countryInfo.

2. Abbreviations of state names in America given by Wikipedia[4].

3. Alternative names mined by pattern matching from the article where the mention appeared. For example, by using the pattern '<mention>, (<abbr>)' we can get the alternate name 'RSA' of mention 'Republic of South Africa' from sentence 'Republic of South Africa, (RSA)'.

Fuzzy matching is enabled since there are some incorrect spellings in source articles which lead to empty results. However, fuzzy matching introduces noises, so it is enable only if the original query recalls nothing.

2.3.2 Candidate Ranking

We formulate the candidate ranking problem as a binary classification problem. Given a mention detected, several potential candidates are retrieved during candidate generation stage. We take every mention and a candidate pair as input for a binary classifier to decide whether the mention refers to the candidate. We consider the classification confidence as the candidate ranking score $score\langle m, e\rangle$ to select the most likely candidate. To deal with context-poor KB problem, we design information rich features and use the ensemble of model strategy.

Features We divide all the features into four groups, i.e., Name String Similarity, Candidate Attributes, Contextual Features and Mention List Features.

1. **Name String Similarity** Following previous work (Shen et al., 2015), we developed features capturing similarity between the candidate's and the mention's name, including Exact Match, Mention Substring of Candidate, Candidate Substring of Mention, Mention Starts Candidate Name, Candidate Starts Mention Name, Jaccard Similarity, Levenshtein Similarity. All names are lowercased in advance and the name of candidate may change into its alternate names if exist.

2. **Candidate Attributes** This set of features are based on target KB's (GeoNames) records and capture some priority of candidate, including Popularity, Number of Ancestors, Code Level.

3. **Contextual Features** Inspired by previous work (Guo et al., 2013), We designed this set of features to measure the contextual similarity between the mention and the candidate. Firstly, for mentions, we take multiple levels of context around mentions in documents as mention-side context, including sentenceparagraph and document level. Secondly, since target KB (GeoNames) lacks context information, we resort to Wikipedia to request candidate's page via API [5]. Considering computation efficiency and avoiding the noise introduced by whole wiki page, we just use the summary (first description paragraph) of the page as candidate-side context, instead of multiple levels. Finally, Bag-of-words representation is employed to mention-side and candidate-side context. Several similarity methods have been explored, including word overlap, cosine similarity and Jaccard similarity.

4. **Mention List Features** We found that the true candidate (or it's ancestor candidates)

[3]https://lucene.apache.org/

[4]https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations

[5]https://github.com/goldsmith/Wikipedia

may also refer to another mention in the same document. This makes sense because toponyms often co-occur with their child or parent toponyms in medical articles or just occur repeatedly in the same document. We developed so called Mention Neighbors Features, which take all mentions in a document as mention list. Similar to mention-side context, every mention has its sentence, paragraph, and document mention list. We encode the relationship between multi-level mention lists and by checking whether the candidate name, its ancestor name or alternate names occur in the mention lists. This set of features can capture the coherence to some extent.

Classification Model We use LightGBM (Ke et al., 2017) as our base model, which gets higher performance compared with other gradient boosting models such as gbdt, xgboost and more traditional models like LR and SVM.

Ensemble & Stacking We select different hyper parameters of LightGBM to build a set of base models. Hyper parameters vary in number of estimators, number of leaves, and learning rate. Furthermore, We add a soft-vote classifier as model ensemble, which returns the class label as argmax of the sum of predicted probabilities. Based on all the base models (several LightGBMs, two vote classifiers), we apply a model stacking strategy that takes the outputs (probabilities and labels) of all base models as input and train a simple linear classifier called stacking model and return the stacking model output as the final output.

3 Experiments

3.1 Toponym Detection

3.1.1 Dataset and Settings

Given 105 medical papers from PubMed Central[6] for developing system, we randomly divided the data into training, development and test set by a ratio of 5:1:1. To avoiding instability of experimental results, we repeat this process 5 times and yield different distributions. All the results shown below are average values among these five distributions.

Data Augment The official training data is smaller than the dataset used in general NER task. Therefore, we expanded the training data by selecting external data from CONLL2003 and ontonotes5.0. Sentence containing GPE or LOC entities were selected. A binary classifier [7] was applied to distinguish the external sentences from the official sentences and outputs a confidence score. If the score lower than a threshold, in other words, the external sentence is similar to the official sentence, we add the external sentence into training data. Finally, we obtained 8000 extra training sentences, about 32% of the total training data.

Preprocessing Articles are segmented into sentences by NLTK and segmentation errors are corrected based on NER results (generated by CoreNLP). For example, ”St. Louis” is split by ’.’ incorrectly. But it is a location according to NER Results.

3.1.2 Ablation Study

Table 1 shows the ablation study of the detection model. As mentioned above, the baseline model is a Char-LSTM-LSTM-CRF model (Lample et al., 2016). We tried two types of pre-trained embeddings, GloVe (Pennington et al., 2014) and PubMed [8]. Since the PubMed is trained on in-domain data, it achieves better results. Thus, all the rest results are based on embeddings trained on PubMed dataset.

Among the four linguistic information, adding ELMo yields the most improvement, while adding the other three yield a little. we successfully use voting, a simple ensemble method to take advantage of multiple models trained by using different linguistic information, and it works.

All techniques proposed contribute to the performance according to the results. Bring in more training data indeed works but the improvement is feeble. Processing tables separately increases the recall since there are many tables containing toponyms.

The best result is obtained by leveraging all the approaches in combination, it outperforms the baseline model significantly.

3.2 Toponym Disambiguation

3.2.1 Dataset and Settings

Data The distributions of articles is the same as those experiments of Toponym Detection. Exter-

[6] `https://www.ncbi.nlm.nih.gov/pmc/`

[7] The training data for this classifier is obtained by mixing the official and external sentences with source information kept. The threshold is chosen by intuition.

[8] trained on PubMed and Wikipedia articles, downloaded from `http://bio.nlplab.org/`

Model		Precision	Recall	F1-score
Baseline+PE	GloVe	85.61	82.81	84.19
	PubMed	87.60	83.24	85.37
Baseline+PE+LF	POS	87.73	83.19	85.40
	NER	87.38	83.55	85.42
	Chunking	87.92	83.47	85.64
	ElMo	89.40	88.34	88.87
Baseline+PE+LF+ME		89.63	88.51	89.06
Baseline+PE+DA		88.25	83.73	85.93
Baseline+PE+TP		88.36	84.78	86.53
Baseline+PE+PP		87.96	83.41	85.62
+All		**90.69**	**89.74**	**90.21**

Table 1: Experiment results of Detection. Abbreviations: DA, Data Augment; PPE, Pubmed Pre-trained Embedding; PP, Post Process; TP, Table Process; LF, Linguistic Features; ME, Model Ensemble

nal data is not included since they contains no annotation for disambiguation task.

Hyper-parameters LightGBM models trained with different hyper-parameters constitute the base model set. The number of estimators varies from 200 to 800, number of leaves from 30 to 50, and the learning rate takes one of 0.05, 0.1. Variance threshold is set as 0.9 at feature selection phase.

3.2.2 Candidate Ranking Results

Table 2 shows the experimental results. We compare the baseline method, single LightGBM model, soft-vote method, and stacking method. The baseline method take the candidate with most population as output.

From Table 2, we can see the LightGBM model beat the baseline method, and model combination strategy improve the performance further. We take outputs of all LightGBM models and soft-vote model as input samples for training a stacking LR model and get the best performance of 89.85%.

For the final run in competition, we chose the stacking method and retrained all base models on the entire train set and predicted on the test set.

3.2.3 Ablation Study

We also conducted an ablation study to investigate the impact of each group of features. From Table 3, we can see Name String Similarity is far below the baseline method(80.45%) and and using the population as a feature is a strong heuristic in fact. Although attribute features take the population as one feature but the classifier using these features still fail to beat the baseline. A reasonable explanation is some other attributes act as noise.

Not surprisingly, Contextual Features play a great role and bring an essential improvement surpassing the baseline. Interestingly, Mention List features, allow a bigger progress over Contextual Features. We think they capture the particularity of toponym disambiguation and some coherence.

Model	Prec.	Rec.	F1
baseline	79.96	80.94	80.45
lightGBM-single	89.30	87.03	88.15
soft-vote	89.44	87.83	**88.63**
stacking	90.57	89.14	**89.85**

Table 2: Main results of Candidate Ranking on entire trainset

	Prec.	Rec.	F1
+name similarity	60.40	63.06	61.70
+ attribute	75.98	76.75	76.36
+ contextual	86.56	85.31	**85.93**
+ mention list	89.30	87.03	**88.15**

Table 3: Ablation study for Ranking features

4 Conclusion and Future works

This paper introduces our system for toponym resolution which is a pipeline of sequence labeling model based detection and classification model based disambiguation. More works are worthy to be done in the future, such as developing a more sophisticated approach for detection toponyms in table, adopting graph-based disambiguation methods and address this task in an end-to-end manner.

References

Wenliang Chen, Yujie Zhang, and Hitoshi Isahara. 2006. Chinese named entity recognition with conditional random fields. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 118–121.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

Stephen Guo, Ming-Wei Chang, and Emre Kiciman. 2013. To link or not to link? a study on end-to-end tweet entity linking. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1020–1030.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Chunping Ma, Huafei Zheng, Pengjun Xie, Chen Li, Linlin Li, and Luo Si. 2018. Dm_nlp at semeval-2018 task 8: neural sequence labeling with linguistic features. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 707–711.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460.

Yushi Yao and Zheng Huang. 2016. Bi-directional lstm recurrent neural network for chinese word segmentation. In *International Conference on Neural Information Processing*, pages 345–353. Springer.

Brenda Starr at SemEval-2019 Task 4: Hyperpartisan News Detection

Olga Papadopoulou, Giorgos Kordopatis-Zilos, Markos Zampoglou, Symeon Papadopoulos, Yiannis Kompatsiaris
Centre for Research and Technology Hellas, Information Technologies Institute, Thessaloniki, Greece
(olgapapa,georgekordopatis,markzampoglou,papadop,ikom)@iti.gr

Abstract

In the effort to tackle the challenge of Hyperpartisan News Detection, i.e., the task of deciding whether a news article is biased towards one party, faction, cause, or person, we experimented with two systems: i) a standard supervised learning approach using superficial text and bag-of-words features from the article title and body, and ii) a deep learning system comprising a four-layer convolutional neural network and max-pooling layers after the embedding layer, feeding the consolidated features to a bi-directional recurrent neural network. We achieved an F-score of 0.712 with our best approach, which corresponds to the mid-range of performance levels in the leaderboard.

1 Introduction

The emerging issue of online disinformation has lately attracted the public attention and is perceived as a major risk for democracy and society. Media content (text, images, videos) is often disseminated on the Internet with the purpose of manipulating public opinion. Hyperpartisan news detection is a problem arising as a result of the intention of publishers to influence readers in favour of a given party, idea or person. The SemEval 2019 Task 4 (Kiesel et al., 2019) seeks solutions to this challenge, in particular text-based approaches that can detect hyperpartisan news articles.

We experimented with two approaches: i) a standard supervised learning approach using superficial text and bag-of-words features, and ii) a deep learning system. We deployed the developed systems on TIRA (Potthast et al., 2019) (a platform that supports software submissions) and its evaluation was conducted on unseen news articles. The results of our submissions, which are presented in Table 1, are promising, yet there is still considerable room for improvement. Our best resulting approach was the deep learning system, which scored an F-score of 0.712. The implemented approaches are described below along with additional experiments that were conducted on the provided training and validation datasets.

2 Data

The dataset provided by the organizers of the task (Kiesel et al., 2019) consists of news articles, half of which are labelled as hyperpartisan. It is split into two sets, the training and the validation set, where for each article the article title, body and published date are provided. The training set consists of 500.000 news articles and it is used as training set for the presented experiments and the provided validation set (150.000 news articles) is used for validating the approaches. A small dataset of 645 news articles, manually annotated, is also provided but not used in the following experiments neither as training nor as validation data. For the evaluation phase, two small datasets of 628 and 4000 articles are provided. The first, called by-article test dataset, is labeled through crowdsourcing on an article basis while the latter, named by-publisher test dataset, is labeled by the overall bias of the publisher as provided by BuzzFeed journalists and MediaBiasFactCheck.com.

A pre-processing step is applied on both the article title and body in order to clean the text and prepare it for the subsequent machine learning steps. The Natural Language Toolkit (NLTK) (Bird et al., 2009) was used to implement this step. First, the text is split into sentences and then each sentence is split in tokens. Lemmatization is applied on each token in order to group together the inflected forms of a word and subsequently remove the stop words based on a list of commonly agreed stop words provided by the NLTK.

	By article test set			By publisher test set		
	Precision	Recall	F-score	Precision	Recall	F-score
SuCla	0.556	0.643	0.596	0.535	0.809	0.644
BOW	0.542	**0.971**	0.696	**0.627**	0.808	0.706
DL	**0.592**	0.895	**0.712**	0.608	**0.860**	**0.712**

Table 1: Evaluation results on the two unseen test sets provided by SemEval-2019 Task 4.

3 Proposed Approach

We experimented with three approaches:

- **SuCla:** a simple classifier based on *superficial* features extracted from the article text (e.g. *number of words*, *contains pronouns*, *number of explanation marks*) and building supervised machine learning models;
- **BOW:** a 'bag-of-words' text classifier;
- **DL:** a deep learning system based on convolutional neural networks (CNN) (LeCun et al., 2015) and recurrent neural networks (RNN) (Medsker and Jain, 1999).

These are further detailed in the next sections.

In the experiments reported here, the training set was used for building the models and the validation set for calculating the evaluation measures: precision, recall and F-score[1]. The decision threshold is set to 0.5 where probabilities ≥ 0.5 indicate hyperpartisan articles and < 0.5 non hyperpartisan. Regarding the submissions to the task through the TIRA platform, training was conducted offline by concatenating the training and validation sets as input and then, the trained models were deployed to TIRA to classify the new, unseen news articles.

3.1 Superficial Features Classifier (SuCla)

This simple approach is an adaptation of the one introduced in (Boididou et al., 2018), which was used to assess the credibility of Twitter posts. We extracted a set of superficial features from the article title, which are a subset of the *tweet-based* features presented in (Boididou et al., 2018). These are listed in Table 2. In (Boididou et al., 2018), further information about the Twitter user who posted the tweet was used, but such information is not available for the article publisher in this task.

We extracted the title-based features on the training and validation sets. The extracted 15-dimensional feature vectors were first normalized in the [0,1] range and then fed to a Radial Basis Function (RBF) kernel SVM. The model parameters were calculated using a grid searching method. The software was deployed to TIRA and evaluated on the unseen articles of the test set. The normalization of test article features was conducted using the scaling parameters computed from the training set. Then, articles were classified as hyperpartisan or not with a score in the [0,1] range: the higher the score the more likely the article is hyperpartisan. The precision, recall and F-measure of this run are presented in Table 1 for the two test sets of unseen articles (by-pyblisher and by-article). The resulting F-scores of 0.596 and 0.644 for the by-article and by-publisher test set respectively indicate that this approach performs better than random but requires more distinctive features to further improve the accuracy.

#	Title-based features
01	Text length
02	Number of words
03	Contains question mark (Boolean)
04	Contains exclamation mark (Boolean)
05	Contains 1st person pronoun (Boolean)
06	Contains 2nd person pronoun (Boolean)
07	Contains 3rd person pronoun (Boolean)
08	Number of uppercase characters
09	Number of positive sentiment words
10	Number of negative sentiment words
11	Number of slang words
12	Has : symbol (Boolean)
13	Number of question marks
14	Number of exclamation marks
15	Number of nouns

Table 2: List of features extracted from the article title.

3.2 Bag-of-words Classifier (BOW)

A text item, in our case the article title or body, can be represented as a vector of word occurrences. This is the well-known and widely used 'bag-of-words' (BOW) model. For building the BOW, we

[1]https://en.wikipedia.org/wiki/Precision_and_recall

	Precision		Recall		F-measure	
	Title	Body	Title	Body	Title	Body
MNB	0.54	0.54	0.66	0.79	0.59	0.65
RF	0.56	0.54	0.74	0.68	0.64	0.60
LR	0.58	0.56	0.79	0.81	**0.67**	0.66

Table 3: Evaluation results for Bag of Words on article title and body. Three classifiers are evaluated: Multinomial Naive Bayes (MNB), Random Forest (RF) and Logistic Regression (LR).

started with the clean text resulting from the pre-processing step described in Section 2 and counted the number of occurrences of each word from two vocabularies that were created based on the training set, and had a size of 64,663 and 364,359 words for the title and the body respectively. Three classifiers were evaluated: a) Multinomial Naive Bayes (MNB), b) Random Forest (RF) and c) Logistic Regression (LR). The obtained test results are presented in Table 3. According to it, LR outperforms the other two, irrespective of whether the article title or body is used as input. The resulting F-scores are 0.67 (title) and 0.66 (body). The BOW counts the number of times a word appear in the text of an article (term frequency) regardless of its appearance in other articles. In addition, we applied the Term Frequency-Inverse Document Frequency (TF-IDF), which adapts the term weight in relation to the times that this term appears in all articles. However, the resulting F-score of 0.58 (title) and 0.66 (body) for LR indicated that classification performance would suffer. Additionally, in the attempt to take advantage of both the title and body text, we implemented a fusion step based on averaging the prediction scores of the individual models. As a result, a minor increase of the F-score to 0.69 was obtained at the expense of additional complexity.

The LR classifier was finally trained on the full set of articles (both training and validation sets) and article title. The new BOW model was deployed to TIRA to classify the unseen news articles. This led to slightly better results as presented in Table 1. Compared to the SuCla approach, the BOW performance is significantly better, especially on the by-article dataset.

3.3 Deep Learning System (DL)

An overview of the employed network architecture, which was devised for the task, is presented in Figure 1.

The input to the network is the vectorized form of the articles' title and body. The input text is pre-processed as described in Section 2. An additional step is applied in order to form the text so that the inputs to the network have the same shape for each article. More specifically, for each article we retain the first 64 sentences, and for each sentence the first 64 words. This results in a (64x64)-dimensional tensor that is provided as input to the network. Zero padding is applied in order to fill missing words and/or sentences.

The input of the network is provided to an Embedding layer, to map each word of the input text to a *word embedding*. We used the pre-trained FastText word embeddings (Mikolov et al., 2018) of size 300. The weights of this layer are not updated during learning. In that way, we overcome the limitation of a bounded vocabulary, imposed by the training set, and the network can process words outside the training sets since they exist in the vocabulary of FastText. The output of this layer is a tensor of (64x64x300) for each article.

Then, we apply multiple convolution filters with different kernel sizes on the output of the Embedding layer. In that way, the network can capture word sequence structure in different granularity levels. The convolutional layers are used with kernel sizes of (1x1), (1x3), (1x5), and (1x7) and in combination with a ReLU activation function. The output of each convolutional layer is a (64x64x128)-dimensional tensor. The outputs of the four convolutional layers are then concatenated on the channel axis (the last tensor dimension) to form a (64x64x512)-dimensional tensor per article. Finally max-pooling is performed over the word axis, i.e., the maximum value per channel and sentence is extracted. To this end, the Embedding and Convolutional layers of the network capture word-level information from the article text.

After max-pooling on the outputs of the Convolutional layers, the (64x512)-dimensional tensors are given to a bidirectional Recurrent Neural Network (bi-RNN) (Schuster and Paliwal, 1997) that

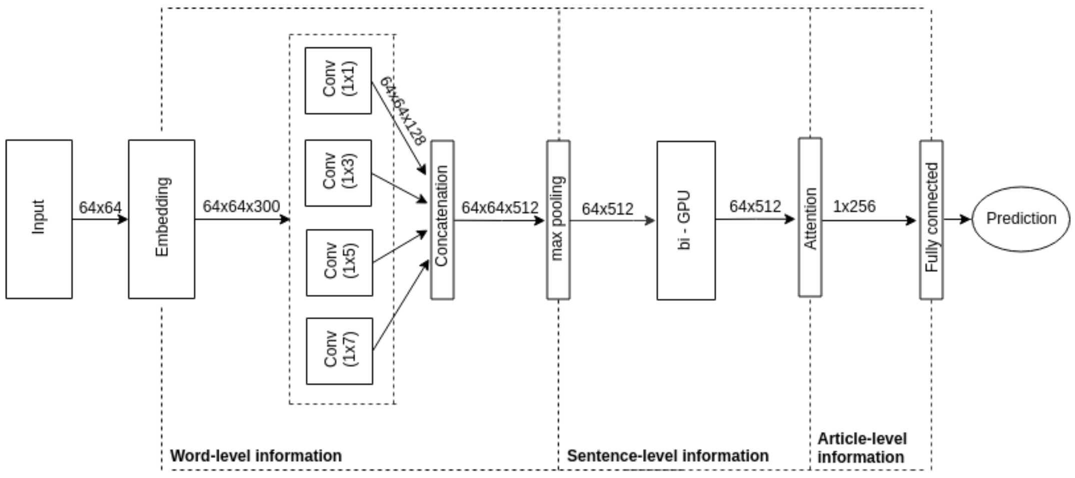


Figure 1: The deep learning architecture developed for classifying a news item as hyperpartisan or not.

calculates sentence vectors by taking into account the neighbor sentences. More precisely, for every article sentence i, the hidden vector h_i summarizes the neighbor sentences around sentence i but still focuses on that sentence. We employed the bidirectional Gated Recurrent Units (bi-GRU) (Cho et al., 2014) as the recurrent unit of the bi-RNN, which is an improved version of the standard recurrent unit. The output of the bi-GRU layer is provided to an attention mechanism (Yang et al., 2016) that weights each sentence vectors based on their similarity to a sentence-level context vector, and then averages the weighted vectors to single vector. The result of the Attention layer is a (1x256)-dimensional vector.

At the final stage, the network captures article-level information. The output of the Attention layer is fed to a fully connected layer to get the final prediction of the network. In this layer, we apply Sigmoid activation to map the output to the [0,1] range, which represents the probability of the article being hyperpartisan. Finally, the network is trained with the binary cross-entropy loss function, weight decay with a $5 * 10^{-4}$ regularization factor, Adam (Kingma and Ba, 2014) optimizer, and 10^{-3} learning rate. Training is done for 100 epochs with a batch size of 32 articles, and the best network is selected based on the performance on the validation set.

This method performs better than the other two approaches, achieving an F-score of 0.712 (Table 1) for both test sets.

3.4 Ideal Fusion

We implemented an ideal fusion method in order to examine the complementarity between the three proposed approaches. This is a theoretical scheme (oracle) which takes the outputs of the individual approaches and selects the correct classifier: at least one model needs to classify correctly an article. An F-score of 0.85 is achieved on the validation set, far better than the individual classifiers accuracy (SuCla: 0.51, BOW: 0.67, DL: 0.65) indicating that the models bring complementary information, which make them good components of a combined model.

4 Conclusions

This paper summarized our participation in SemEval-2019 Task 4, where we aimed at the challenge of Hyperpartisan News Detection. We tried to approach the problem from the perspective of standard supervised learning techniques, as well as more complex deep learning approaches. While none of the methods gave groundbreaking results, our set of experiments and observations provides a solid basis for future research on the problem. In particular, we intend to conduct more extensive analysis on the annotated data and extract patterns that will be more representative and distinctive for the problem at hand. Moreover, we will consider combining the three proposed approaches with the aim of creating a stronger and more accurate combined model. The significant increase in performance of the ideal fusion method points out the benefits of such a strategy.

5 Acknowledgments

This work is supported by the WeVerify project, which is funded by the European Commission under contract number 825297.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Christina Boididou, Symeon Papadopoulos, Markos Zampoglou, Lazaros Apostolidis, Olga Papadopoulou, and Yiannis Kompatsiaris. 2018. Detection and visualization of misleading content on twitter. *International Journal of Multimedia Information Retrieval*, 7(1):71–86.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*, 521(7553):436.

Larry Medsker and Lakhmi C Jain. 1999. *Recurrent neural networks: design and applications*. CRC press.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. TIRA Integrated Research Architecture. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Cardiff University at SemEval-2019 Task 4: Linguistic Features for Hyperpartisan News Detection

Carla Pérez-Almendros, Luis Espinosa-Anke and Steven Schockaert
School of Computer Science and Informatics
Cardiff University, UK
{perezalmendrosc,espinosa-ankel,schockaerts1}@cardiff.ac.uk

Abstract

This paper summarizes our contribution to the Hyperpartisan News Detection task in SemEval 2019. We experiment with two different approaches: 1) an SVM classifier based on word vector averages and hand-crafted linguistic features, and 2) a BiLSTM-based neural text classifier trained on a filtered training set. Surprisingly, despite their different nature, both approaches achieve an accuracy of 0.74. The main focus of this paper is to further analyze the remarkable fact that a simple feature-based approach can perform on par with modern neural classifiers. We also highlight the effectiveness of our filtering strategy for training the neural network on a large but noisy training set.

1 Introduction

In the era of misinformation, the challenge of differentiating reality from frames, facts from opinions, is becoming increasingly important. Concepts such as *Fake News*, *Fact Checking* or *Post-Truth Era*, generally unknown a few years ago (Lewandowsky et al., 2017), started to play an important part in media, academic papers and even in Natural Language Processing (NLP) tasks (Rashkin et al., 2017; Shu et al., 2017; Wang, 2017). Nowadays, strongly opinionated news stories can offer biased information, as is the case with hyperpartisan articles. A text is considered hyperpartisan when it is highly polarized towards an extreme position. Potthast et al. (2018) analyzed hyperpartisanism in relation to fake news, to discover that a very similar writing style could be associated both with right-wing and left-wing polarized stories. This shared style of biased articles was different from that of mainstream articles. Task 4 in SemEval 2019 (Kiesel et al., 2019) consisted in a classification challenge where news articles had to be sorted out as hyperpartisan or non-hyperpartisan. Our approach addressed this challenge via two different models, which included 1) an SVM classifier based on word embeddings averages and handcrafted linguistic features and 2) a recurrent BiLSTM neural network classifier trained on filtered data. The reason and process for filtering data will be explained in section 3.2.

Word embeddings have remained central to the state-of-the-art approaches in NLP since the introduction of Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) models. In addition to modelling word meaning, embeddings can also be applied to longer units of significance, such as phrases, sentences, paragraphs or entire documents (Le and Mikolov, 2014). However, although such vector representations often enable the best results in a variety of NLP challenges, some tasks still benefit from linguistically inspired approaches. In fact, recent work has proved how linguistic features and stylometry could improve the performance of deep learning techniques. The already mentioned work in Hyperpartisan and Fake News detection (Potthast et al., 2018) applied stylometry based on linguistic features to identify strongly biased articles. Bag of words, stop words, part of speech and readability scores are some of the features analyzed by the authors. They also focus on quotes, measuring their length and counting their appearances in a text. Rhetorical questions or the appearance of personal pronouns, among many others linguistic features, also helped to classify suspicious vs trusted news posts on Twitter (Volkova et al., 2017). The number of adverbs or swear words has also been used for fact checking purposes (Rashkin et al., 2017). Inspired by these previous works and their results, these linguistic features are some of the ones we apply in our first model.

The second model is based on word embeddings as input for a neural network. Specifically, we used a Convolutional Neural Network (CNN) combined with a Bidirectional Long-Short Term Memory (BiLSTM) network. The main novelty of this approach lies in the preprocessing step which filters the training data. This strategy is used because the bulk of the training data only provides a weak supervision signal, which we found too noisy to use directly.

2 Data

The data provided for this task consisted of a training set of 645 news articles, manually labeled as hyperpartisan or non-hyperpartisan. In addition to this gold standard data, a larger dataset of 600,000 training articles and 150,000 validation articles was provided. These complementary documents were labeled automatically depending on their source media. If the media was considered hyperpartisan, the article was labeled as such, but without analyzing its content. These automatically tagged articles also included further labels (referring to the publisher rather than the article itself), which have not been used in our system. Our first model exclusively relied on the manually labeled articles (645) for training, while the second model also took advantage of the larger set of weakly labeled articles (750000).

3 Our Approach

The dataset was preprocessed by changing the text to lower case and then applying tokenization. For our first model, sentence tokenization was applied and articles were preprocessed with part-of-speech tagging (PoS), using the NLTK library (Bird et al., 2009). While Potthast et al. (2018) kept and analyzed quotes in texts, we chose to delete them in both datasets. Our first experiments showed that, in a small number of cases, keeping quotes drove our system to misclassification because while an article could quote hyperpartisan statements of others, the document itself did not necessarily have an extreme position towards a topic or event. In both approaches, we only considered the main text of the article, discarding its title.

3.1 Model 1: Combining Document Embeddings with Linguistic Features

Document embeddings were built for each article. For doing so, we first computed embeddings for all sentences by averaging the pre-trained GloVe vectors (Pennington et al., 2014) for all the words occurring in them. To this end, we used the uncased Common Crawl pretrained GloVe embeddings, with 300 dimensions and a vocabulary of 1.9 million words. The average of all the sentences in an article was then computed to obtain a single vector representation of the news article. To complement this document vector, we identified a number of document-level discriminant linguistic features to classify a text as hyperpartisan or non-hyperpartisan. The selected features are as follow:

- **`excl`**: total number of exclamation marks
- **`quest`**: total number of question marks
- **`adj`**: percentage of tokens which are adjectives
- **`adv`**: percentage of tokens which are adverbs
- **`insults`**: total number of insults or swear words[1]
- **`first_pers`**: total number of times that the first person personal pronoun *I* was used
- **`sent_length`**: average length of sentences
- **`min_sent`**: length of the shortest sentence
- **`max_sent`**: length of the longest sentence

These feature values were concatenated with the document vector to provide the input for a linear SVM classifier.

To experiment with different configurations of our method, we used the 645 manually labeled articles with 5-fold cross-validation. Document embeddings and linguistic features vectors were considered both separately and concatenated as input for different classifiers, namely Random Forest, Logistic Regression and Support Vector Machine (SVM), with different parameters. In all cases, we used the implementations from the Scikit-Learn Machine Learning Library (Pedregosa et al., 2011). After testing the different options, a concatenated vector of document embeddings and linguistic features as input for an SVM proved to ob-

[1] A file containing swear words and insults was provided as input for a swear words count function. The file was created with a list of such words extracted from https://www.digitalspy.com/tv/a809925/ofcom-swear-words-ranking-in-order-of-offensiveness/, and then augmented with 2,500 similar insults coming from their *word2vec* nearest neighbours.

tain the best results. We finally trained our combined model as a linear SVM on the entire set of 645 gold standard documents.

3.2 Model 2: Neural Text Classification

Neural networks need large amounts of data to be able to learn. The small dataset of 645 manually labeled articles was clearly too small to train a competitive neural network model. However, we noticed that the large training set of 750K documents, which was labeled based on the publisher, was too noisy, yielding a performance which was far worse than that of the first model. We attempted to surmount this issue via a two-step strategy, in which we first trained a classifier on the small training set, which we used to filter the larger but noisy training set. The goal was to automatically extract from the 750K labeled-by-publisher articles only those which were correctly predicted as hyperpartisan or non-hyperpartisan by this initial classifier. The strategy which we found to perform best was the following:

1. Using half of the 645 labeled articles, we trained three classifiers:
    - a linear SVM based on the linguistic feature set described in Section 3.1,
    - a linear SVM based on the document embedding
    - and a standard neural classifier using a convolutional layer followed by a bi-LSTM layer (CBLSTM).

    The two first classifiers were included after they had been tested in our first model, and with the parameters previously explained. The choice of using a combination of CNN and LSTM for our third classifier stems from previous work where either or both architectures combined proved to be useful in document classification (Kim, 2014; Xiao and Cho, 2016). Concerning the choice of hyperparameters, we used 100 convolutional filters of size 5, and one-token strides. The output of the CNN layer was then passed to a max-pooling layer (where pool size was set to 4), and this output was passed to a bidirectional LSTM layer which produces two 100d vector outputs, which after concatenation, were passed to a final 2d softmax layer. We implemented this model using the *keras*[2] library.

2. We then trained a meta-classifier on the remaining half of the 645 articles, which used the predictions of these three individual classifiers as features, and which finally generated a final prediction. For this meta-classifier we used a linear SVM.

3. Once trained, the meta-classifier was applied to the 750K labeled-by-publisher articles. Whenever the ground-truth label agreed with the prediction of our metaclassifier, the article was retained in our filtered dataset.

4. Through this process, we obtained around 150K refined labeled articles that we used for training another CBLSTM, replicating the same process and parameters used in step 1. This last refined model was the one applied to the test set.

4 Analysis

We will focus our analysis on the first model, given that its result is perhaps most surprising. We observed that exclamation and question marks were present in non-hyperpartisan and mainstream articles, but a high occurrence of these features is nonetheless clearly correlated with hyperpartisanism. Adjectives and adverbs tended to be more frequent in hyperpartisan texts as well. Insults or swear words were extremely scarce in non-hyperpartisan articles, so their presence is a strong indicator for hyperpartisanism. The personal pronoun *I* denotes a personal text, and for this reason is more common in strongly opinionated articles. Average sentence length was not found to be particularly informative. On the other hand, we found that the shortest sentences in hyperpartisan articles tend to be shorter than those in non-hyperpartisan articles. In a similar way, the longest sentences were also slightly longer in extreme news stories. These results have been summarized in Table 1.

Further experiments to assess the discriminatory power of each linguistic feature were performed, although these took place after the submission for this SemEval task. A 5-fold cross validation on the 645 gold-standard articles dataset was applied to estimate the performance of the model in each case. As can be seen in Table 2, the linguistic features on their own are sufficient for achieving an accuracy of 66%. Combining them with document embeddings, the results reached 73.2%. However, a deeper analysis showed that

[2]https://keras.io/

	Hyp.	Non Hyp.
excl (avg)	1.30	0.63
quest (avg)	2.43	1.20
adj (avg)	9.00	8.40
adv (avg)	4.10	3.20
insults (avg)	0.07	0.01
first_pers (avg)	3.26	1.78
sent_length (avg)	22.47	24.43
min_sent(median)	2.00	4.00
max_sent(median)	52.00	47.00

Table 1: Linguistic features extracted from 645 articles dataset.

some linguistic features were actually deteriorating the general performance of the system. For instance, the linguistic features model alone performed better when `excl` were not accounted for. On the other hand, `insults`, `adj` and `adv`, respectively, were the most discriminant features, leading to the biggest drop in performance when discarded. Here, we would like to highlight that, surprisingly, the feature `adv` reduces the performance of the combined model, where eliminating it allows the system to reach 75% of accuracy. The feature `first_person` is also reducing the score of the combined system. However, whenever we omitted two or more linguistic features, the performance of the combined model dropped below 72.7%, which is the accuracy achieved by the document vectors on their own. Therefore it seems safe to conclude that our linguistic features share some information that, combined, provide complementary evidence for document embeddings.

	Ling. Feat.	Comb. model
complete model	**.660**	**.732**
- excl	.666	.738
- quest	.663	.730
- adj	.642	.730
- adv	.651	.750
- insults	.640	.738
- first_pers	.662	.742
- sent_length	.660	.735
- min_sent	.662	.730
- max_sent	.662	.736
only document embeddings	-	**.727**

Table 2: Ablation results for the first model in terms of accuracy.

5 Results and Discussion

Both our systems obtained around 74% accuracy in SemEval 2019 task 4: Hyperpartisan News Detection under the team name of Ankh-Morpork Times (Potthast et al., 2019). This constitutes an improvement of 28 percentage points over the provided baseline. In the SemEval competition, our team got the 16th position out of 42 participants. Our main contribution was to show that a simple approach, based on document embeddings and linguistic features, can obtain the same accuracy as a typical neural text classifier.

Overall, there are several lessons learned from our participation in this task, which we will try to develop in future work. For example, we confirmed that, although the community tends to rely on the performance of word vectors, linguistic features can complement word vector based representations in a meaningful way in text classification. In addition, further analysis in our work showed that we could have improved our performance with a better selection of linguistic features. Therefore, for future work, we aim at providing a more reliable model which takes into account more complex linguistic features. For instance, we believe that looking at sentences' modality and sentiment, as well as assessing the polarity of adjectives and adverbs in a text, should give valuable extra information for the task.

As a secondary contribution, we also proposed a technique for filtering noisy data. It is known that neural networks perform well for large training sets, but sometimes a large accurately labeled dataset cannot be obtained. To this end, we created a meta-classifier trained on a smaller gold standard dataset and applied to larger, noisy data for obtaining a filtered higher-quality training set.

6 Namesake

Ankh-Morpork is the biggest city in the Discworld, the fictional world that gives name to the famous fantasy book series by Sir Terry Pratchett. And Ankh-Morpork Times is its first, biggest and most famous newspaper, and covers in a peculiar and surreal way the no less surreal events happening in this flat world. And sometimes, we must admit, with quite a hyperpartisan point of view.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Stephan Lewandowsky, Ullrich KH Ecker, and John Cook. 2017. Beyond misinformation: Understanding and coping with the post-truth era. *Journal of Applied Research in Memory and Cognition*, 6(4):353–369.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. TIRA Integrated Research Architecture. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 231–240. Association for Computational Linguistics.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2931–2937.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, 19(1):22–36.

Svitlana Volkova, Kyle Shaffer, Jin Yea Jang, and Nathan Hodas. 2017. Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 647–653.

William Yang Wang. 2017. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprint arXiv:1602.00367*.

Clark Kent at SemEval-2019 Task 4: Stylometric Insights into Hyperpartisan News Detection

Viresh Gupta, Baani Leen Kaur Jolly, Ramneek Kaur, Tanmoy Chakraborty
Indraprastha Institute of Information Technology, Delhi (IIIT-Delhi), India
{viresh16118, baani16234, ramneekk, tanmoy}@iiitd.ac.in

Abstract

In this paper, we present a news bias prediction system, which we developed as part of a SemEval 2019 task. We developed an XGBoost based system which uses character and word level n-gram features represented using TF-IDF, count vector based correlation matrix, and predicts if an input news article is a hyperpartisan news article.

Our model was able to achieve a precision of 68.3% on the test set provided by the contest organizers. We also run our model on the BuzzFeed corpus and find XGBoost with simple character level N-Gram embeddings to be performing well with an accuracy of around 96%.

1 Introduction

The problem of hyperpartisan news detection (Potthast et al., 2018) is based on predicting whether a news article is biased towards a specific political wing or not. The problem falls under the category of classification problems, and the task is to classify an article as being extremely one-sided or not. A closely related problem is that of fake new detection wherein the task is to analyze the veracity of an article, and classify it based on some predefined degrees of truthfulness.

Our problem has a high societal relevance, since one-sided news poses a great threat to democracy, particularly in the context of conducting fair elections. In this paper, we discuss our approach to solving this problem used during the contest *Hyper Partisan News Detection*, a competition task at SemEval 2019 (Kiesel et al., 2019).

More formally, our problem definition is:

Definition 1 (Hyperpartisan News Detection) *We are given a set of news articles A, where each article a_i is marked with two labels: a Boolean label hyperpartisan h_i which indicates if article a_i is biased towards a political wing, and a bias label $b_i \in$ {left, right, left-center, right-center, least} which indicates which wing the article is biased towards. If h_i = True, then $b_i \in$ {left, right}; if h_i = False, then $b_i \in$ {least, left-center, right-center}. The objective is to learn a classifier **C** which predicts the hyperpartisan label h_j for an unknown news article a_j.*

In this work, we identify the role of various traditional NLP features in determining the degree of partisanship. We utilise standard term-frequency and inverse document frequency vector features computed for uni, bi and tri-grams obtained from the corpus. We do this feature extraction at both character and word level and then train a gradient boosted decision tree as a classifier for identifying partisanship. We also compare other methods of classification such as SVM, KNN, Naive Bayes and Logistic Regression for the task using the same vector features. Furthermore, experiments exploiting the metadata information were also performed (explained in detail in the scalar features in section 3.2).

The experiments were performed on two corpora, the BuzzFeed corpus (created in (Potthast et al., 2018)) and the training corpus released by the task organisers (the SemEval corpus). Further we also discuss the results obtained on the final test corpus released for the final evaluation of the task in section 4.1. Due to computation infeasibility over the larger training corpus, we do not compute vector features for the SemEval corpus.

2 Related Work

The work done on hyperpartisan and fake news detection can be broadly classified into three categories – knowledge-based (Etzioni et al., 2008; Ginsca et al., 2015), context-based (Long et al.,

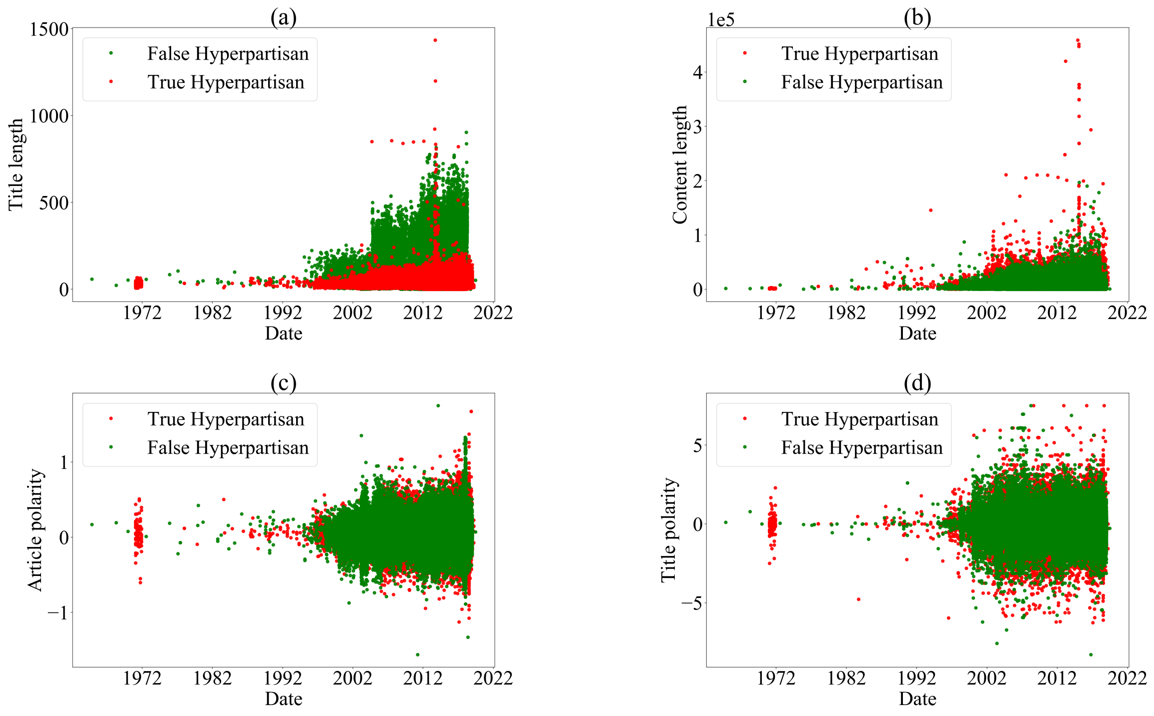


Figure 1: Variation in feature value w.r.t. time for true and false hyperpartisan articles in SemEval Corpus (a) Title length w.r.t. time (b) Content length w.r.t. time (c) Article polarity value w.r.t. time (d) Title polarity value w.r.t time . Red and green colors depict articles from hyperpartisan publishers and other publishers respectively.

2017a; Mocanu et al., 2015), and style-based (Bourgonje et al., 2017).

While the knowledge-based and context-based features may take some time to detect hyperpartisanship (after the news starts spreading on social media), the style-based features can be used to detect partisanship of a news article well in time before the damage happens (Potthast et al., 2018).

For exploiting style based features, (Long et al., 2017b) uses deep learning based methods, and (Shu et al., 2017) performs fake news detection on social media data using a data mining oriented approach.

2.1 Baseline

We take as our baseline the work done by (Potthast et al., 2018). Their work uses the author's writing style as features to detect hyperpartisanship. The stylometric features used in their work include POS-unigrams, POS-bigrams, POS-trigrams, char-unigrams, char-bigrams, char-trigrams, stopword-uniGrams, stopword-bigrams, stopword-trigrams, general inquirer categories, readability scores, quotation ratio, link amount and average paragraph length. A random forest classifier was used to make predictions.

We use their classifier as the baseline for the BuzzFeed corpus. For the SemEval corpus, we use the random baseline provided in the task as our baseline. The baseline results are mentioned in Tables 1 and 3 for both the datasets.

3 Methodology

In this section, we describe the dataset, the features that we selected and the models we trained using the selected features. A visual overview is shown in Figure 2.

3.1 Corpus

We used two corpora, which we name as BuzzFeed corpus and SemEval corpus.

BuzzFeed corpus: This corpus was produced by the baseline work. The dataset comprised 1,627 articles that were manually checked by four BuzzFeed journalists. Of these, 826 articles belong to the main-stream category of publishers, 256 belong to the left-wing category of publishers, and the remaining 545 to the right-wing category of publishers.

SemEval corpus: This corpus has been released for the SemEval 2019 Task 4 on Hyperpartisan News Detection. It comprises 800,000

Baseline Results				
Model	**Precision**	**Recall**	**F1 score**	**Accuracy**
RF	0.75	0.77	0.75	0.75
Count Vector Results				
Model	**Precision**	**Recall**	**F1 score**	**Accuracy**
XGB	0.92	0.93	0.92	0.93
LR	0.92	0.92	0.92	0.93
SVM	0.89	0.90	0.89	0.91
KNN	0.75	0.78	0.76	0.76
GNB	0.75	0.77	0.73	0.73
RF	0.71	0.70	0.62	0.62
Word N-gram Vector Results				
Model	**Precision**	**Recall**	**F1 score**	**Accuracy**
XGB	**0.95**	0.95	**0.95**	**0.96**
SVM	0.89	0.91	0.91	0.91
LR	0.87	0.90	0.88	0.89
GNB	0.82	0.85	0.82	0.82
RF	0.74	0.73	0.64	0.64
KNN	0.72	0.70	0.62	0.61
Character N-gram Vector Results				
Model	**Precision**	**Recall**	**F1 score**	**Accuracy**
XGB	**0.95**	**0.96**	**0.95**	**0.96**
SVM	0.89	0.91	0.90	0.91
GNB	0.87	0.89	0.88	0.89
RF	0.87	0.89	0.87	0.89
LR	0.85	0.88	0.85	0.86
KNN	0.67	0.57	0.40	0.43

Table 1: Vector feature results (BuzzFeed Corpus only).

	Precision	**Recall**	**F1 score**	**Accuracy**
RF	**0.81**	0.64	**0.72**	**0.74**
LR	0.63	0.76	0.69	0.65
KNN	0.62	0.66	0.64	0.62
GNB	0.57	**0.93**	0.71	0.61
SVM	0.52	0.90	0.66	0.58

(a) BuzzFeed Corpus.

	Precision	**Recall**	**F1 score**	**Accuracy**
KNN	**0.64**	0.59	0.62	**0.63**
RF	0.55	0.76	0.64	0.58
SVM	0.62	0.08	0.15	0.52
GNB	0.51	**0.94**	**0.66**	0.51
LR	0.48	0.57	0.52	0.47

(b) SemEval Training Corpus.

Table 2: Scalar features results.

training articles and 200,000 test articles. These articles are annotated based on the publisher of the articles.

3.2 Feature Selection

Prior to the selection of features, we pre-processed our datasets to clean the text in articles to handle the encoding errors, perform text normalisation and stop word removal. The features we selected can be categorized into two categories, viz. scalar features and vector features. We train two sets of models, one for each category of features.
Scalar features: Here, we select four features, all used at the same time since they encode different information:

- Article length: This feature denotes the length of the articles in terms of the number of characters.
- Title length: The title length features denotes the length of the title of an article in terms of the number of characters.
- Article polarity: The article polarity denotes the sentiment score of the article text in the range $[-1, 1]$. A score value less than zero implies a negative sentiment, and a positive sentiment otherwise.
- Title polarity: Similar to the article polarity, the title polarity feature denotes the sentiment score of the article title in the range $[-1, 1]$.

Vector features: These include three kinds of features (considered separately since they encode the same information):

- Word count vectors: The count vector for a document denotes the vector of counts of words in the document from the set of possible words in a corpus/vocabulary.
- Word level n-gram vectors: The word level vector for a document denotes the vector of tf-idf values of words level n-grams in the document. We used unigrams, bigrams and trigrams for this feature.
- Character level n-gram vectors: The character vector for a document denotes the vector of counts of character level ngrams. For this feature too, we use unigrams, bigrams and trigrams.

Visual inspection of the data: In Figure 1 we provide a visual insight into the corpus based

Dataset	Method	Precision	Recall	F1 score	Accuracy
By Article	Ours	68.3	17.8	28.3	**54.8**
	Baseline	46.2	46.0	44.3	45.1
By Publisher	Ours	56.5	17.0	26.1	**51.95**
	Baseline	51.1	51.1	50.0	50.5

Table 3: Results for the submitted model.

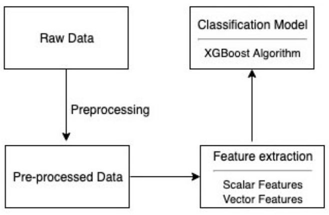


Figure 2: System Overview.

on the features selected. The figure depicts scatter plots showing variation in feature values w.r.t. time for both true and false hyperpartisan articles.

3.3 Models Used

We use the following learning models for our scalar features of the BuzzFeed corpus: K Nearest Neighbours (KNN), Gaussian Naive Bayes (GNB), Random Forest (RF), Logistic Regression (LR) and Support Vector Machine (SVM). For the vector features of the BuzzFeed corpus, we use: KNN, GNB, RF, LR, SVM and XGBoost (XGB).

4 Experiments

We divide this section into three parts – experimental setup, results on the BuzzFeed corpus, and results on the SemEval corpus.

4.1 Experimental Setup

The article polarity and title polarity features were computed using SentiWordNet[1] (Baccianella et al., 2010). All the vector features were computed using the scikit-learn package. To split the data into training and testing sets, we used 5-fold cross-validation.

4.2 Results on the BuzzFeed Corpus

The results for the scalar features for models trained on the BuzzFeed corpus are shown in Table 2a(a). The RF model performs the best with an accuracy of 74%. The scalar features, however, are insufficient in beating the baseline. We therefore train models on our vector features. The results for the vector features are shown in Table 1.

As evident from table 1 and 2a(a), vector features perform much better and are able to beat the baseline (Table 1) easily.

Various sections in Table 1 represent the results when using different kinds of vector representations as features, with character level n-grams yielding the top results.

4.3 Results on the SemEval Corpus

Results on the SemEval corpus are shown in Table 2b(b). From all models, KNN performs the best, followed by RF, SVM, and GNB (in that order).

Since computing vector features and tf-idf features was computationally infeasible on this corpus, we did not train the vector features, however, based on our observations from buzzfeed dataset (i.e the character level vectors outperforming all others), we trained a supervised classifier using FastText (Joulin et al., 2016), (Bojanowski et al., 2016). The accuracy achieved for this model is 65%.

The results of our model using all the scalar features on the final evaluations (testing by article and testing by publisher corpus) of this competition are shown in Table 3. These results show that our model suffered from the inability to draw out more of the relevant results (low recall).

5 Conclusion

In this work, we have explored traditional sets of features and models for the Hyper-partisan News Detection problem. We worked on two corpora, of which one has been used in the state-of-the-art literature. For this corpus, we beat the baseline and achieve a remarkable accuracy of 96%. For the other corpus, we achieve an accuracy of 65% (with a fast text character level embedding based model).

From the results of the contest (Table 3), we were able to beat the baseline easily. Though our system did not achieve as high accuracy as other systems, we observe that this is due to a bad recall, i.e even though the features that we selected

[1]https://github.com/anelachan/sentimentanalysis

are very useful for the model to produce relevant results, it cannot capture some of the correct results.

6 Code and Reproducibility

We provide all our code for both Buzzfeed and Semeval Corpus as a github repository located at https://github.com/virresh/hyperpartisan-semeval19-task4 . The same code was uploaded on TIRA (Potthast et al., 2019) and run for submission to the contest.

Acknowledgement

Part of the research was supported by the Ramanujan Fellowship, Early Career Research Award (SERB, DST), and the Infosys Centre for AI at IIITD.

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. volume 10.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv e-prints*, page arXiv:1607.04606.

Peter Bourgonje, Julian Moreno Schneider, and Georg Rehm. 2017. From clickbait to fake news detection: An approach based on detecting the stance of headlines to articles. In *Proceedings of the 2017 EMNLP Workshop: Natural Language Processing meets Journalism*, pages 84–89. Association for Computational Linguistics.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. 2008. Open information extraction from the web. *Commun. ACM*, 51(12):68–74.

Alexandru L. Ginsca, Adrian Popescu, and Mihai Lupu. 2015. Credibility in information retrieval. *Foundations and Trends in Information Retrieval*, 9(5):355–475.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of Tricks for Efficient Text Classification. *arXiv e-prints*, page arXiv:1607.01759.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Yunfei Long, Qin Lu, Rong Xiang, Minglei Li, and Chu-Ren Huang. 2017a. Fake news detection through multi-perspective speaker profiles. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 252–256. Asian Federation of Natural Language Processing.

Yunfei Long, Qin Lu, Rong Xiang, Minglei Li, and Chu-Ren Huang. 2017b. Fake news detection through multi-perspective speaker profiles. In *IJCNLP*.

Delia Mocanu, Luca Rossi, Qian Zhang, Marton Karsai, and Walter Quattrociocchi. 2015. Collective attention in the age of (mis)information. *Computers in Human Behavior*, 51:1198 – 1204. Computing for Human Learning, Behaviour and Collaboration in the Social and Mobile Networks Era.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. TIRA Integrated Research Architecture. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 231–240. Association for Computational Linguistics.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake News Detection on Social Media: A Data Mining Perspective. *arXiv e-prints*, page arXiv:1708.01967.

Dick-Preston and Morbo at SemEval-2019 Task 4: Transfer Learning for Hyperpartisan News Detection

Tim Isbister
Swedish Defence Research Agency
Stockholm, Sweden
tim.isbister@foi.se

Fredrik Johansson
Swedish Defence Research Agency
Stockholm, Sweden
fredrik.johansson@foi.se

Abstract

In a world of information operations, influence campaigns, and fake news, classification of news articles as following hyperpartisan argumentation or not is becoming increasingly important. We present a deep learning-based approach in which a pre-trained language model has been fine-tuned on domain-specific data and used for classification of news articles, as part of the SemEval-2019 task on hyperpartisan news detection. The suggested approach yields accuracy and F1-scores around 0.8 which places the best performing classifier among the top-5 systems in the competition.

1 Introduction

In today's polarized media and political landscapes, the challenge of determining whether a news article is biased or not is highly topical. In the hyperpartisan news detection task (Kiesel et al., 2019) of the International Workshop on Semantic Evaluation (SemEval) 2019, the task is to predict whether a given news article text follows a hyperpartisan (extreme one-sided) argumentation or not, i.e., whether it exhibits blind or prejudiced allegiance to one party, cause, or person (Potthast et al., 2019). As part of this challenge, participating research teams got access to two datasets:

1. **by-publisher**: A well-balanced dataset consisting of 750,000 articles in which the data have been labeled by the overall bias of the *publisher*, as provided by journalists or fact-checking sites.

2. **by-article**: A smaller dataset consisting of 645 articles for which crowdsourcing workers have agreed on the labeling of the *articles* as being hyperpartisan (37%) or not (63%). A similar but more well-balanced test dataset (to which the participating teams have not got direct access) has been used for evaluating the accuracy, precision, recall, and F1-score of systems developed by the participating research teams.

In this system description paper we present the results for the two participating research teams from the Swedish Defence Research Agency (FOI): 1) **dick-preston** and 2) **morbo**.

The teams contributed with separate systems for the early-bird deadline and for the final submission. In the early phase we used traditional machine learning classifiers such as logistic regression and support vector machines (SVMs), built upon traditional text features such as word and character n-gram term frequencies (weighted with inverse document frequency). These classifiers have been used as baselines to which more "modern" NLP classifiers have been compared. For the final submission both teams made use of transfer learning-based Universal Language Model Fine-Tuning (ULMFiT) models. The difference in the teams' final systems is the percentage of data used for training/validation splits when fine-tuning the models and the number of epochs for which the models were trained. Despite that only a few hundred examples were used for fine-tuning the pre-trained ULMFiT-models, accuracies and F1-scores of approximately 0.8 were achieved on the unseen test data. This resulted in a fifth place for the team **dick-preston** and seventh place for the team **morbo** out of 42 participating teams, as reported on the competition leaderboard[1].

The rest of this paper is structured as follows. In Section 2, we present the machine learning algorithms and features which have been used for building the hyperpartisan news article classifiers used in the competition. In Section 3 we outline

[1]https://pan.webis.de/semeval19/semeval19-web/leaderboard.html

the conducted experiments, present the used hyperparameters, and describe the obtained results. Finally, we present overall conclusions and discuss ideas for future work in Section 4.

2 Method

In the early phase of the competition, both FOI teams experimented with traditional machine learning algorithms such as Naïve Bayes, logistic regression, and support vector machines (SVMs), taking sparse text features such as word and character n-grams as input. These methods have been used as baselines to which more novel algorithms have been compared. The FOI baseline methods are briefly presented in Section 2.1.

For the final system submission we have used more "modern" NLP methods. More specifically, Universal Language Model Fine-Tuning (ULMFiT) was utilized. ULMFiT is a natural language processing (NLP) transfer learning algorithm introduced in (Howard and Ruder, 2018). ULMFiT is one of several language model-based transfer learning algorithms developed in 2018 which have been shown to yield state-of-the-art results on several NLP tasks. Approaches such as ELMo (Peters et al., 2018), OpenAI GPT (Radford et al.), and BERT (Devlin et al., 2018) have arguably received more attention than ULMFiT, but we selected to implement our final systems using ULMFiT due to its straightforward implementation in the fastai library[2], and its promising results also on small datasets (Howard and Ruder, 2018). ULMFiT is presented in more detail in Section 2.2.

2.1 Baseline Classifiers

As baseline classifiers we have made use of traditional "shallow" machine learning algorithms like logistic regression, SVMs, etc. An extensive list of the tested algorithms can be found in the experiment descriptions in Section 3. A detailed explanation of such classifiers is outside the scope of this paper but we refer the interested reader to (Hastie et al., 2001) for an excellent introduction to such approaches.

As input features to our baseline classifiers we have used term frequencies of n-grams. In the most basic case of 1-grams (unigrams), this means that for each token in the dataset (tested on character as well as word level) we count the number of times the specfic token (e.g., the word "Trump") appears. In the case of 2-grams (bigrams) we do the same, but then for pairs of tokens (e.g., "President Trump"). To account for tokens which appear frequently in all kinds of news articles (thereby making them less valuable for prediction of the target class) we weigh the term frequencies by their inverse document frequency. Various strategies such as only including the most frequently occuring tokens have also been utilized. Details of which strategies that have been tested in our experiments are given in Section 3.

2.2 ULMFiT

As the basis of our ULMFiT models we have used a pre-trained language model trained on the English Wikitext-103 (Merity et al., 2016), which in total consists of more than 28,000 preprocessed Wikipedia articles and over 100 million words. The pre-trained language model consists of a word embedding layer connected to a three-layered unidirectional left-to-right AWD-LSTM (Merity et al., 2017). The AWD-LSTM utilizes several regularizations strategies such as a DropConnect mask on the hidden-to-hidden recurrent weights and variable length backpropagation through time (BPTT) sequences. Given a sequence of N tokens, a left-to-right language model can be used to compute the probability of the sequence of tokens:

$$P(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} P(t_k | t_1, t_2, \ldots, t_{k-1}) \tag{1}$$

Language models are powerful in that they can "teach themselves" a lot about language by simply letting them iteratively predict the next word in a sequence on large amounts of (otherwise unlabeled) training data. Throughout this process, the parameters in the ULMFiT AWD-LSTM layers implicity learn about both syntax and semantics as these are helpful for predicting the next word in a sequence.

In next step, the pre-trained language model has been fine-tuned on the 645 articles in the manually crowdsourced **by-article** dataset. The reason for this fine-tuning is that the news articles most likely stem from a different data distribution, compared to the Wikipedia articles on which the language model originally have been trained. During the language model fine-tuning, discriminative learning and slanted triangulated learning rates (SLTR) was used, as outlined in the original

[2]https://github.com/fastai/fastai

ULMFiT paper (Howard and Ruder, 2018). The language model could most likely have been improved upon more by making use of the larger **by-publisher** dataset. However, we were interested in how good the ULMFiT model would perform on a very limited dataset.

In the last step, the fine-tuned language model has been augmented with two linear blocks separated by a rectified linear unit (ReLU) activation function. The last linear block consists of just two output nodes with a softmax activation, giving as output a probability of the current news article being hyperpartisan or not given the fine-tuned model. Regularization in the form of dropout and batch normalization is applied to the linear blocks in order to allow for better generalization. Moreover, gradual unfreezing is used in order to avoid catastrophic forgetting, a phenomenon which previously has been common when trying to fine-tune pre-trained language models.

3 Experiments and Results

For the first part of the competition we experimented with baseline classifiers to which we later on could compare the classification accuracy of more advanced algorithms on hold-out validation datasets constructed from the training data. The experiments with the baseline classifiers were performed using scikit-learn, while latter experiments have been carried out using various deep learning frameworks (including TensorFlow and Keras). The final ULMFiT classifier implementations and experiments have been carried out using PyTorch and the fastai library.

3.1 FOI Baseline Classifier Experiments

We first experimented with a number of simple classifiers which were used as baselines:

- SVM (LinearSVC)
- Logistic Regression
- Random Forest
- Gradient Boosting
- Naïve Bayes
- NBSVM

We used scikit-learn to conduct a grid-search over various hyperparameters for these classifiers in order to find suitable optimized baseline models. In this section we will focus on the hyperparameters of the SVM classifier and its input features as this performed the best among the evaluated classifiers.

A consistent result for all the tested classifiers was that they performed better when creating the n-gram features described in last section from the text context of the news articles rather than only using the shorter titles. As input to the classifier we combined the 1000 word unigrams and bigrams ranked highest in terms of TF-IDF and the 1000 character unigrams and bigrams ranked highest in terms of TF-IDF. For the SVM we used a linear kernel and the regularization parameter C was set to 0.38. Using these parameters we obtained a weighted F1-score of 0.78 when applying stratified 10-fold cross validation on the training data. When the same model was trained on 100 % of the training data and submitted for evaluation on the test data as part of the early-bird deadline we obtained an accuracy of 0.77. This is a rather strong baseline as it would have resulted in a top-10 result in the final leaderboard (if the final FOI classifiers would not have been submitted).

3.2 FOI ULMFiT Classifier Experiments

The embedding layer of our ULMFiT classifiers uses word embeddings with an embedding size of 400. For the sequential linear layers that have been attached to the pre-trained LSTM layers we have used a momentum of 0.1 for the BatchNorm and a dropout probability p of 0.2 for the first linear layer and 0.1 the last linear layer. We have gradually unfreezed different blocks of the model to avoid catastrophic forgetting. Different slanted learning rates and number of training epochs have been used for the different submitted FOI classifiers, but we have in general found learning rates around 0.01 to work well for fine-tuning just the last layer, and then using lower magnitude learning rates when unfreezing earlier layers.

We evaluated the fine-tuned ULMFiT classifiers by splitting the available **by-article** dataset into a training set (85 %) and a validation set (15 %). This was attempted on a few random splits for which we consistently reached accuracies on the validation set over 0.95. In the end we submitted models trained on 85 % and 100 % of the training data but the one trained on 85 % performed the best, probably due to overfitting of the other model (which is natural since it was hard to know how

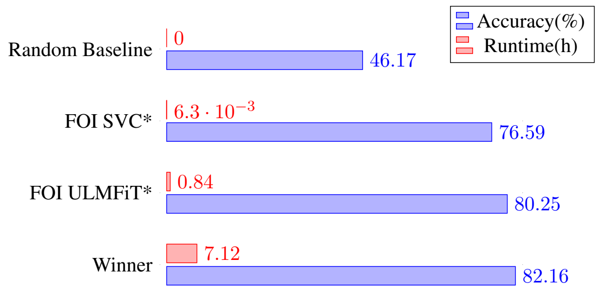


Figure 1: Accuracy and runtime of FOI classifiers in comparison to a random baseline and the winner of the hyperpartisan news detection competition.

many epochs the model should be trained for when not having any separate validation data to evaluate on). When the best performing model was submitted for evaluation on the test set it obtained an accuracy of 0.80 which resulted in a fifth place in the final leaderboard.

In Figure 3.2 we compare the accuracy and running time of our best performing ULMFiT classifier on the test data and contrast them to the corresponding measures for our FOI linear SVM classifier, a random baseline provided by the task organizers, and the classifier developed by the winning team. As can be seen, the accuracy of the ULMFiT classifier is marginally lower than the winning classifier, while the running time seems to be much lower[3].

4 Conclusions

In this paper we have described the ULMFiT classifiers developed by the FOI teams **dick-preston** and **morbo** for the SemEval-2019 challenge of hyperpartisan news article detection. By first fine-tuning a pre-trained language model on the texts and titles of a small dataset consisting of 645 news articles and then fine-tuning two additional linear blocks on humanly annotated labels of these articles, we have achieved accuracy and F1-scores around 0.80 on the task organizers' test dataset. The obtained accuracies resulted in a fifth and seventh place, respectively, out of a total of 42 research teams who submitted their classifiers to the competition. This demonstrates the applicability of novel transfer learning approaches such as ULMFiT to domains for which only very limited amounts of data is available. To the best of our knowledge, this is the first time on which ULMFiT has been attempted on such a small dataset.

[3]The submissions were evaluated on a rather slow virtual machine (Potthast et al., 2018) which impact the running times.

4.1 Future Work

The obtained results could have been improved upon by utilizing the larger available **by-publisher** training set for improving the fine-tuning of the language model on the target domain. It is also possible that this larger dataset could have been used for further fine-tuning of the classifier.

Another interesting idea for future research on this dataset would be to train a classifier based on Google AI's BERT, which make use of a deep bidirectional transformer instead of a multi-layered LSTM.

Acknowledgments

This work was supported by the R&D programme of the Swedish Armed Forces.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2001. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing LSTM language models. *CoRR*, abs/1708.02182.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR*, abs/1609.07843.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. TIRA Integrated Research Architecture. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 231–240. Association for Computational Linguistics.

Alec Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. Available: https://blog.openai.com/language-unsupervised/.

Doris Martin at SemEval-2019 Task 4: Hyperpartisan News Detection with Generic Semi-supervised Features

Rodrigo Agerri
IXA NLP Group, University of the Basque Country UPV/EHU
rodrigo.agerri@ehu.eus

Abstract

In this paper we describe our participation to the Hyperpartisan News Detection shared task at SemEval 2019. Motivated by the late arrival of Doris Martin, we test a previously developed document classification system which consists of a combination of clustering features implemented on top of some simple shallow local features. We show how leveraging distributional features obtained from large in-domain unlabeled data helps to easily and quickly develop a reasonably good performing system for detecting hyperpartisan news. The system and models generated for this task are publicly available.

1 Introduction

The definition of hyperpartisan according to the Hyperpartisan News Detection shared task at SemEval 2019 (Kiesel et al., 2019) is the following: "Given a news article text, decide whether it follows a hyperpartisan argumentation, i.e., whether it exhibits blind, prejudiced, or unreasoning allegiance to one party, faction, cause, or person".[1] Putting it simply, the task is, given a news article, to decide whether such document is hyperpartisan (true) or not (false). This task is related to the Stance Detection (Mohammad et al., 2016) and automatic detection of fake news (Pérez-Rosas et al., 2018) tasks, which are getting increasing attention within the Natural Language Processing community (Potthast et al., 2018). In this sense, it could be the case that hyperpartisanism is conveyed by some elements of fake news within the article, usually with the objective of spreading propaganda and manipulate readers towards a particular stance on a specific topic.

The SemEval 2019 task 4 aims to address the problem of hyperpartisan news detection at document level, without trying to distinguish specific elements or indicators of hyperpartisanism in each article. Two sets of data were released to participants. The first part (*bypublisher*) is annotated at publisher level. This means that if a publisher is thought to be spreading hyperpartisan news, then all its articles are annotated as hyperpartisan. The *bypublisher* set contains 750K articles divided in 600K documents for training and a validation set of 150K documents. The second part (*byarticle*) has been annotated at article level via crowdsourcing and consists of 645 articles for training and 628 documents for the test. The test set is hidden in TIRA (Potthast et al., 2019) and it is used for the official evaluation scores of the task. It should be noted that, unlike the *byarticle* test set, the *byarticle* training set was not balanced (407 false vs 238 true).

We address this task using an existing document classification system, mostly due to the fact that we joined the task just a week before the final submission deadline. However, and despite the lack of time to implement specific features for the task, we obtained quite good results with a simple and very general feature set in which the most meaningful feature was the use of pre-trained clusters obtained from the English Wikipedia and the Gigaword 5th edition. Out of 42 participants, our official submission obtained 0.737 accuracy whereas the winner of the task scored 0.822.

In addition to our official participation, in this paper we also describe a second round of experiments performed after the official submission deadline. The objective was to establish whether using clusters trained on domain-specific data would improve the results with respect to those obtained by using clusters based on general domain text such as Wikipedia and Gigaword. As it turned out, this second round of experiments allowed us to considerably improve the results

[1] https://pan.webis.de/semeval19/semeval19-web/index.html

(0.761) with respect to our official scores in the task (0.737), confirming that training clusters on domain-specific data, although smaller, helps to address the hyperpartisan news detection task.

2 Methodology

We parsed the given data in XML format extracting the title and the document body for training. We experimented with the original corpus version and with a cleaned (HTML tags removed) and tokenized version. All the pre-processing was done using the IXA pipes tools (Agerri et al., 2014).

Our system learns language independent models which consist of a set of local, shallow features complemented with semantic distributional features based on clusters obtained from a variety of out-of-domain and domain-specific data sources. We show that our approach, despite the lack of hand-engineered, language- and task-specific features, obtains competitive results in the hyperpartisan news detection task.

For the official results we trained only on the *byarticle* training set. The best settings of our system were chosen via 5-fold cross validation. The chosen models and software were uploaded to TIRA (Potthast et al., 2019) to annotate and evaluate the test data. For the official runs, we used pre-trained clusters from the Wikipedia and the English Gigaword, as described by Agerri and Rigau (2016).

For the second round of experiments, we used the large *bypublisher* data set and a Fake News Kaggle set[2] in order to train clusters. The motivation was to test whether using data sources closer to the task domain, as opposed to using general text data from Wikipedia and Gigaword, helped to obtained better word representations for this task.

3 ixa-pipe-doc

Our document classification system is *ixa-pipe-doc*, which aims to establish a simple and shallow feature set, avoiding any linguistic motivated features, with the objective of removing any reliance on costly extra gold annotations (POS tags, lemmas, semantics) and/or cascading errors if automatic language processors are used. The underlying motivation is to obtain robust models to facilitate the development of document classification systems for several languages, datasets and domains while obtaining state of the art results.

[2] https://www.kaggle.com/c/fake-news

The system consists of: (i) Local, shallow features based mostly on orthographic, word shape and n-gram features plus their context; (ii) three types of simple clustering features, based on unigram matching; (iii) publicly available gazetteers, such as sentiment lexicons. Specifically, *ixa-pipe-doc* implements, on top of the local features, a combination of word representation features: (i) Brown (1992) clusters, taking the 4th, 8th, 12th and 20th node in the path; (ii) Clark (2003) clusters and, (iii) Word2vec (Mikolov et al., 2013) clusters, based on K-means applied over the extracted word vectors using the skip-gram algorithm. The implementation of the clustering features looks for the cluster class of the incoming token in one or more of the clustering lexicons induced following the three methods listed above. If found, then we add the class as feature. The Brown clusters only apply to the token related features, which are duplicated.

ixa-pipe-doc, as a component of IXA pipes, includes a simple method to combine various types of clustering features induced over different data sources or corpora. This method has already obtained state of the art results in several tasks such as newswire Named Entity Recognition (Agerri and Rigau, 2016) and Opinion Target Extraction (Agerri and Rigau, 2019), both in out-of-domain and in-domain evaluations.

Clusters of words provide denser document representations. Although still a one-hot vector representation, the dimensions of the representation gets reduced to the number of clustering classes used. This is done by mapping the words in the document to the words in each of the clustering lexicons thereby obtaining a denser representations than the traditional one-hot representation based bag of words (Turian et al., 2010).

Finally, *ixa-pipe-doc* learns supervised models via the Maxent algorithm (Ratnaparkhi, 1999). To avoid duplication of efforts, the system uses the Apache OpenNLP project implementation of Maxent [3] customized with the features described in this section.

4 Experiments

We train ixa-pipe-doc with the default parameters, performing 100 iterations with a 5 count cutoff.[4]

[3] http://opennlp.apache.org/

[4] Only features that occur more than 5 times are considered (Ratnaparkhi, 1999).

Features	F1 True	F1 False	Accuracy
token	0.655	0.810	0.755
char26	0.643	0.806	0.749
pref04	0.662	0.810	0.757
token + pref04	0.669	0.809	0.758
token + char26	0.652	0.807	0.752
pref04 + char	0.655	0.812	0.757
(local) Token + char26 + pref04	0.665	0.813	0.759
local + CW600	0.672	0.814	0.763
local + W2VG200	0.674	0.817	**0.766**
local + CW600+W2VG200	0.671	0.816	**0.764**

Table 1: 5-fold cross validation for official results on the *byarticle* training set. CW600: Clark Wikipedia 600 clusters; W2VG200: Word2vec Gigaword 200 clusters.

Features	Accuracy	P	R	F1
Local + W2VG200	**0.737**	0.754	**0.704**	**0.728**
Local + CW600+W2VG200	0.714	**0.773**	0.608	0.680

Table 2: Official results on TIRA test set. CW600: Clark Wikipedia 600 clusters; W2VG200: Word2vec Gigaword 200 clusters.

We only tested three types of **local features** which were already implemented in the system: the current token, the character ngrams of each token (2:6 range) and word prefixes (0-4 characters of each token).

Due to our late arrival to the task, we combined the best local features with our pre-trained clusters from Wikipedia and Gigaword for the official results described in section 4.1. For the second round of experiments of section 4.2, we used the clusters trained using the *bypublisher* and Fake News datasets. The number of clusters trained with each algorithm and data source was the following: 100-800 clusters using the Clark and Word2vec methods, and 1000 classes with the Brown algorithm. The best combination of features were obtained by performing every possible permutation between them in a 5-fold cross validation setting using the *byarticle* training data.

4.1 Official Results

Table 1 provides the 5-fold cross validation results used to choose the two best runs that we submitted for testing on TIRA. As it can be seen, the performance for the *true* and the *false* classes greatly differ. This could be due to the unbalanced nature of the *byarticle* training set or because classifying articles that are hyperpartisan is actually more difficult.

The official results obtained by our system are shown in Table 2. These results show that the main weakness of the system is its lower recall. The local features used usually obtain high precision and lower recall whereas the clustering features reduce sparsity thereby improving the recall. The exception was the Brown clusters, which were detrimental to performance. This is consistent with previous experiments using clusters trained in out-of-domain data (Agerri and Rigau, 2019). Finally, although TIRA did not show the results per class (true or false) we believe that our system reproduced, for the official test data, the behaviour observed in the cross validation experiments.

Therefore, our results seem to indicate that the data used in our pre-trained clusters, Wikipedia and Gigaword, does not allow us to create good word representations for the hyperpartisan news data. Still, it can be said that our official results were promising, obtaining 0.737 versus the 0.822 accuracy of the best system.[5]

4.2 Second Round

This second round of experiments consisted of replacing the out-of-domain cluster lexicons from Wikipedia and Gigaword with those trained on the *bypublisher* and Fake News data. These are the "local + clusters" models in Table 3, which shows the results of performing 5-fold cross validation

[5]https://pan.webis.de/semeval19/semeval19-web/leaderboard.html

Features	F1 True	F1 False	Accuracy
local + W2VHP300	0.675	0.819	0.769
local + W2VFN400	0.670	0.813	0.761
local + W2VHP300+W2VFN400 (clusters)	0.677	0.825	0.773
local + clusters + polarity	0.675	0.824	0.772
local-token + clusters	0.677	0.826	0.774
local-token + clusters + polarity	**0.678**	**0.827**	**0.775**

Table 3: 5-fold cross validation for the second round of results on the *byarticle* training set. W2VHP300: Word2vec Hyperpartisan bypublisher 300 clusters; W2VFN400: Word2vec Fake News 400 clusters.

Features	Accuracy	P	R	F1
local	0.707	**0.768**	0.592	0.669
local + W2VHP300+W2VFN400 (clusters)	0.754	0.719	0.834	0.772
local + clusters + polarity	0.754	0.717	**0.840**	**0.774**
local-token + clusters	0.756	0.731	0.808	0.768
local-token + clusters + polarity	**0.761**	0.734	0.818	0.774

Table 4: Second round results. W2VHP300: Word2vec Hyperpartisan bypublisher 300 clusters; W2VFN400: Word2vec Fake News 400 clusters.

on the *byarticle* training set in order to choose the best models for testing.

Furthermore, Table 3 reports the results of three additional experiments: (1) adding three polarity lexicons to the *local* + *clusters* model; (2) removing the current token feature from the *local* feature set (*local-token* + *clusters*) and, (3) adding the three polarity lexicons to experiment (2). The motivation of removing the current token feature was to see if that helped the system to generalize better over unseen words. The features based on polarity add a polarity value (positive or negative) if a word in training or testing gets matched in one of the three polarity lexicons used. More specifically, we used three different lexicons (Hu and Liu, 2004; Riloff and Wiebe, 2003; Mohammad et al., 2009), resulting in three different features for each token. As in the previous section, in this phase we realized that our system consistently performs much better, for every experiment, for the "false" class. Experimenting with a balanced training set is left for future work.

As we expected, using domain-specific clustering-based word representations substantially improved the recall results, which in turn led to substantial improvements in terms of accuracy and F1 score. This improvements are reflected also on the evaluation on the test data hidden in TIRA. Thus, Table 4 reports considerable gains obtained by using clustering features in terms of recall with respect to the model based on local features only. The final reported score is 0.761 in accuracy, still lower than the top score in the task (0.822), but a significant result obtained by the simple method of providing better word representations (closer to the task domain) based on clustering. The improvements of this second round of experiments are larger in terms of F1 score, which goes up to 0.774, closer to the winner's F1 score of 0.809.

Most importantly, our experiments show that our system, even though generic, simple and lacking task-specific features, allows to easily obtain competitive results for a document classification task such as hyperpartisan news detection.

5 Concluding Remarks

This paper describes our first experiments on the Hyperpartisan News Detection task organized at SemEval 2019 (Kiesel et al., 2019). We aim to improve our work in the task by using other techniques such as denser word representations based on continuous vectors (word embeddings) and deep learning architectures for document classification. We would also like to investigate the relation with other tasks such as Stance Detection (Mohammad et al., 2016) and automatic detection of fake news (Pérez-Rosas et al., 2018). The system and models can be found in `https://github.com/ixa-ehu/ixa-pipe-doc`.

Acknowledgments

This work has been supported by Spanish Ministry of Economy and Competitiveness (MINECO/FEDER, UE), under the project CROSSTEXT (TIN2015-72646-EXP) and the Ramon y Cajal Fellowship RYC-2017-23647.

References

Rodrigo Agerri, Josu Bermudez, and German Rigau. 2014. IXA pipeline: Efficient and ready to use multilingual NLP tools. In *Proceedings of the 9th Language Resources and Evaluation Conference (LREC2014)*.

Rodrigo Agerri and German Rigau. 2016. Robust multilingual named entity recognition with shallow semi-supervised features. *Artificial Intelligence*, 238:63–82.

Rodrigo Agerri and German Rigau. 2019. Language independent sequence labelling for opinion target extraction. *Artificial Intelligence*, 268:85–95.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 59–66. Association for Computational Linguistics.

M. Hu and B. Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

S. Mohammad, C. Dunne, and B. Dorr. 2009. Generating high-coverage semantic orientation lexicons from overtly marked words and a thesaurus. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 599–608.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41. Association for Computational Linguistics.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic detection of fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3391–3401. Association for Computational Linguistics.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. TIRA Integrated Research Architecture. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 231–240. Association for Computational Linguistics.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine learning*, 34(1-3):151–175.

E. Riloff and J. Wiebe. 2003. Learning extraction patterns for subjective expressions. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'03)*.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden. Association for Computational Linguistics.

Duluth at SemEval-2019 Task 4: The Pioquinto Manterola Hyperpartisan News Detector

Saptarshi Sengupta and **Ted Pedersen**
Department of Computer Science
University of Minnesota
Duluth, MN 55812, USA
{sengu059,tpederse}@d.umn.edu

Abstract

This paper describes the Pioquinto Manterola Hyperpartisan News Detector, which participated in SemEval–2019 Task 4. Hyperpartisan news is highly polarized and takes a very biased or one–sided view of a particular story. We developed two variants of our system, the more successful was a Logistic Regression classifier based on unigram features. This was our official entry in the task, and it placed 23rd of 42 participating teams. Our second variant was a Convolutional Neural Network that did not perform as well.

1 Introduction

Social media has become a vital source of news for many people. It makes it possible to share useful information widely and in a timely fashion, and yet can also be misused to spread biased, misleading, or dangerous content.

Hyperpartisan news is a particular worry in that it is premised on absolute allegiance to one particular point of view, and seeks to reinforce potentially misinformed opinions held by its readers. This has led to very real consequences in this world. A tragic example can be found in Myanmar, where Buddhist ultranationalists relied on social media to spread hyperpartisan and fake news in order to promote hatred and violence against different Muslim communities (Fink, 2018).

While related, Hyperpartisan news is not the same as fake news. The former shows a high degree of bias, whereas the latter is more so an outright fabrication. However, the techniques applied to detecting both are similar. For example, Pérez-Rosas et al. (2018) detected fake news by training Support Vector Machines using ngrams, punctuation, and measures of readability. (Tacchini et al., 2017) used *likes* of articles as features for building a Logistic Regression classifier for fake news detection. Potthast et al. (2018) identified hyperpartisan news through the use of style and readability features, and also employed a technique known as *unmasking* (Koppel et al., 2007) to distinguish between hyperpartisan and mainstream news.

2 Task Description

SemEval–2019 Task 4 (Kiesel et al., 2019) challenged participants to detect whether an article is hyperpartisan (H) or mainsteam (M). As such it represents a binary classification task. The task organizers provided training data, and so we elected to take a supervised learning approach.

There were two datasets provided by the organizers (Kiesel et al., 2018). The *by-article* data is a smaller corpus of 645 news articles that have been manually assigned to H (238 articles) or M (407 articles). There was also the much larger *by-publisher* data set with 750,000 articles where classifications were made based on the source of an article. Making classifications in this way is possible since certain publishers are known to be providers of hyperpartisan content. For our experiments we elected to use the by-article data, but plan to investigate the potential of the by-publisher data in future work.

3 Methodology

We created two systems for the task.[1] The first was a Logistic Regression (LR) classifier trained on unigram features, and the second a Convolutional Neural Network (CNN) with word embeddings created from the training data.

During the development phase of our systems we carried out 10-fold cross validation on the *by-article* training data in order to tune both our LR and CNN systems.

[1] https://github.com/saptarshi059/SemEval2k19-Task4-UMD

3.1 Logistic Regression

Logistic Regression (LR) is a widely used method for supervised learning. Each feature is assigned a positive or negative weight which indicates the contribution of that feature to the overall classification of the system. We carried out our experiments using *scikit-learn* (Pedregosa et al., 2011), a Machine Learning toolkit for Python.

Our first step was to preprocess the text. This consisted of converting all text to lowercase, and removing stopwords and non-alphanumeric characters.

Next, a word by article matrix was generated for the training data. For our purposes words are defined as space separated strings. Any word that occurred less than 12 times in the training data was removed and not considered a feature. We arrived at this cutoff via our cross validation experiments, where this value led to the most accurate results (although other nearby values were nearly as accurate).

Our LR model was trained using the default settings for scikit-learn. We relied on the default *liblinear* algorithm (Fan et al., 2008) to optimize the loss function, since it is known to be effective with smaller amounts of training. data[2]

3.2 Convolutional Neural Network

Our initial focus was on our LR approach. However, the task allowed for two entries per team, and so we decided to include a CNN given its history of success in text classification tasks (e.g., (Liu and Wu, 2018)). We used *keras* (Chollet et al., 2015), a Python toolkit for Deep Learning that provides a wrapper around TensorFlow.

Our CNN approach was also based on unigrams, although each unigram was represented by an embedding created from the training data. We started with an existing CNN for text classification[3] and made a few adjustments to some of the hyperparameters. The maximum input vector size was set to 10,000, our embeddings were of length 100, and we trained our model for 100 epochs. We used *Adam* to optimize the loss function and a *GlobalMaxPooling1D* layer to reduce the size of the input feature vectors.

We did not experiment with these hyperparameters extensively, but instead relied on what we found to be fairly common settings and defaults provided by *keras*.

[2]https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[3]https://realpython.com/python-keras-text-classification/

Model	**Accuracy**	**P**	**R**	**F1**
LR	0.70	0.74	0.63	0.68
CNN	0.58	0.87	0.18	0.30

Table 1: Final Evaluation Results.

4 Experimental Results

The formal task evaluation was carried on virtual systems provided by the organizers using the TIRA system (Potthast et al., 2019). We trained both our LR and CNN on the entire by-article training corpus and saved the resulting models to disk (so they could be ported over to the evaluation system).

We decided to use LR and CNN as our two entries to the task, since during our development phase they had very similar results on 10-fold cross validation : LR accuracy was 0.77 ± 0.06 while CNN was at 0.75 ± 0.05.

However, on the official evaluation run (using a held out set of test data the systems had never seen), the CNN performed poorly and only attained accuracy of 0.58. LR on the other hand reached accuracy of 0.70 and so was selected by the organizers as our official entry to the task. Other evaluation metrics including Precision (P), Recall (R), and F1 are shown in Table 1.

The confusion matrix for our LR system is shown in Table 2 and for the CNN system in Table 3. In these matrices the distribution of correct or gold standard answers are shown in the columns (with sums 314) and the system predictions are shown across in the rows. While the evaluation phase test data is balanced between the classes H and M, the by-article training data was not (238 H versus 407 M).

Table 2 shows that LR predicted a somewhat more balanced distribution of classes (266 H vs. 362 M), which is reflected in the relatively similar Precision and Recall scores found in Table 1. However, Table 3 shows that the CNN produced a much more skewed result (67 H vs. 561 M) which led to very high Precision for the CNN (0.87) while the Recall was extremely low (0.18).

We hypothesize that the difference between the distribution of classes in the training versus evaluation data at least partially explains this result. Given more examples of mainstream news (M),

	H	M	
H	197	69	266
M	117	245	362
	314	314	628

Table 2: LR Confusion Matrix.

	H	M	
H	58	9	67
M	256	305	561
	314	314	628

Table 3: CNN Confusion Matrix.

both models learned this class more thoroughly and so tended to classify articles into this category.

The LR model appears to be more robust in that it performed at approximately the same level of accuracy both during development phase cross validation and the final evaluation round (despite the difference in the distribution of classes).

The CNN on the other hand appears to have been very negatively affected by the shift in the distribution of classes from training to evaluation data, and performed significantly worse on the evaluation data as compared with cross validation on the training set. We are uncertain as to the causes of the CNN result. It is important to note that the by-article data is relatively small and that this may put the CNN at a disadvantage. We also noticed that the accuracy of the CNN on the training data was 1.00 and much lower on the evaluation data, which is a common sign of overfitting.

5 Feature Analysis

An appealing quality of Logistic Regression is that it is somewhat transparent and allows us to see which features are contributing more to classification decisions. Table 4 shows the top 30 features for LR based on the weights learned from the training data. Positive weights are associated with the hyperpartisan (H) class, and negative weights indicate the mainstream (M) class. We've put the words with negative weights in upper case to improve readability, however remember in our data that all text was lower cased.

While the highly weighted individual features are of interest, it is important to remember that Logistic Regression performs classification based on the combined weight of all the features present in an article. As a result a single highly weighted feature for one class may be overridden by the presence of multiple lesser weighted values for the other class.

Hyperpartisan		Mainstream	
sponsored	.603	**DONALD**	-.611
women	.489	ISIS	-.610
americans	.473	TOOK	-.500
change	.471	SATURDAY	-.417
proud	.463	TWITTER	-.414
hillary	.459	THINK	-.393
arpaio	.440	**WATTERS**	-.392
racist	.436	WORLD	-.389
someone	.433	CLAIMS	-.372
outrage	.423	RUN	-.353
mexican	.373	WEDNESDAY	-.351
threat	.371	ASKED	-.348
political	.370	FREEDOM	-.343
democracy	.369	BORDER	-.334
planned	.366	VIDEO	-.333
supremacist	.364	CONVENTION	-.332
clintons	.337	STATES	-.326
department	.335	ELECT	-.321
use	.329	DEBATE	-.321
desperate	.329	PAST	-.320
originally	.329	**SESSIONS**	-.316
killer	.323	MORNING	-.316
certainly	.322	SAID	-.313
conservative	.320	COUNTY	-.311
father	.313	FOX	-.310
fine	.302	ADVERTISEMENT	-.310
hitler	.302	CONTINUE	-.308
wants	.302	UNITED	-.303
maria	.301	BUSINESS	-.303
make	.299	PRISON	-.301

Table 4: Top 30 LR features : positive weights associated with H, negative with M.

The data for this task consists of articles from 2016 – 2018, starting around the time of the 2016 US presidential election, where Donald Trump defeated Hillary Clinton after a bitterly contested campaign.

In general the top features contain many terms associated with elections or political figures. We note a few more person names among the top 30 features for the H class (5) versus the M class (3). These features are in bold face in Table 4. It is significant to note that one of the person names that appears as a hyperpartisan feature is Hitler, suggesting that he may have been used as a basis for comparison in such articles. The name Arpaio refers to a controversial sheriff in Arizona who ran for re-election in 2016 (and was defeated). Maria is Hurricane Maria, which devastated Puerto Rico in September 2017. The recovery from this natural disaster became a political issue and so its use as a feature in hyperpartisan news seems likely.

The mainstream features (in upper case) include Donald and Twitter. Candidate (and now President) Trump is well known as an enthusiastic Twitter user, so these features would certainly occur in mainstream news coverage. Jesse Watters is a Fox News reporter who hosts a person on the street style interview program which drew some news coverage. Jeff Sessions was an early supporter of Donald Trump and became Attorney General after the election and so was often in the news.

6 Error Analysis

We divided the by-article training data into a set of 585 training examples and 60 test instances (30 from each class). We used this data to train and evaluate our LR classifier. We categorized our results as True Positive (H classified as H), True Negative (M classified as M), False Positive (M classified as H), and False Negative (H classified as M). Below we discuss an article from each category, where each is identified via (by-article id number, word count).

True Positive (1, 259): This article takes a mocking and sarcastic tone regarding President Trump's campaign promises to fix infrastructure. It points out that Hurricane Maria (H feature) did extensive damage but that Trump was indifferent because Puerto Rico did not vote for him. This is an obvious example of hyperpartisan news.

True Negative (14, 225): Ivana Trump, Donald's ex-wife, talks about his punctuality in his personal and professional life. The article is very matter of fact and simply describes her observations without embellishment or bias, and is pretty clearly mainstream.

False Positive (4, 929): This is a very long article that was classified as H despite not having any obvious signs of bias. Rather it compares the unsettled state of America now with the very turbulent year of 1968. However, the article uses many rare and emotional words such as *nihilism*, *malady*, and *hysteria* which may have caused it to be classified as hyperpartisan.

False Negative (2, 189): This is a highly opinionated response to Joyce Newman's (Democrat) stance on gun control. It is a very emotional piece, however, it also provides facts and figures to bolster the position of the author. We believe it is the latter which caused the LR to (incorrectly) classify it as mainstream.

We also noticed that the 30 H articles in our test data had on average much larger word counts (1,178.9) versus the 30 M articles (503.4). (Potthast et al., 2018) used average paragraph length as a feature when detecting H news, and this seems like it would have been a useful feature in this task as well.

7 Future Work

There are numerous possible directions for future work. We are interested in exploring the use of the much larger by-publisher training data. This could be of particular assistance in improving the results from CNNs. We also plan to revisit our preprocessing steps and perform named entity recognition since proper nouns represent important information for this problem.

We would also like to explore variations in our feature sets for LR. In our current experiments we do not have any requirement that a feature occur in a certain minimum number of articles (in addition to occurring at least 12 times). As a result we noticed several features that occurred many times in just a few articles were strongly weighted and yet would be unlikely to generalize well. We would also like to explore the use of *TF-IDF* in place of simple frequency counts for feature selection.

Finally, our error analysis suggested that hyperpartisan news tends to use emotional language as well as unusual or rare words. Given this we are interested in the possibilities offered by sentiment analysis, as well as the inclusion of structural and style features.

8 Namesake

Pioquinto Manterola is a fictional journalist created by Paco Ignacio Taibo II. He is a central character in *The Shadow of a Shadow* (Ignacio Taibo II, 1991) and *Returning as Shadows* (Ignacio Taibo II, 2003). These novels are set in Mexico City, the first in 1922 and the second in 1941-1942. In both stories Manterola is teamed with a poet to investigate mysterious circumstances that lead to uncovering even more complex and sinister wrongdoing. As such he seemed an appropriate namesake for our team in this task.

References

François Chollet et al. 2015. Keras. https://keras.io.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Christina Fink. 2018. Dangerous speech, anti-muslim violence, and facebook in myanmar. *Journal of International Affairs*, 71(1.5):43–52.

Paco Ignacio Taibo II. 1991. *The Shadow of the Shadow*. Viking Books, New York City.

Paco Ignacio Taibo II. 2003. *Returning as Shadows*. Thomas Dunne Books, New York City.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, David Corney, Payam Adineh, Benno Stein, and Martin Potthast. 2018. Data for PAN at SemEval 2019 Task 4: Hyperpartisan News Detection.

Moshe Koppel, Jonathan Schler, and Elisheva Bonchek Dokow. 2007. Measuring differentiability: Unmasking pseudonymous authors. *Journal of Machine Learning Research*, 8:1261–1276.

Yang Liu and Yi-fang Brook Wu. 2018. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 354–361.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic detection of fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3391–3401. Association for Computational Linguistics.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. TIRA Integrated Research Architecture. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 231–240. Association for Computational Linguistics.

Eugenio Tacchini, Gabriele Ballarin, Marco L. Della Vedova, Stefano Moret, and Luca de Alfaro. 2017. Some like it hoax: Automated fake news detection in social networks. *CoRR*, abs/1704.07506.

Fermi at SemEval-2019 Task 4: The sarah-jane-smith Hyperpartisan News Detector

Nikhil Chakravartula[1,3] Vijayasaradhi Indurthi[1,2], Bakhtiyar Syed[2],
[1] Teradata, [2] IIIT Hyderabad
[1]{nikhil.chakravartula,vijayasaradhi.indurthi}@teradata.com
[2]{vijaya.saradhi, syed.b}@research.iiit.ac.in
[3]{nikhil.chakravartula}@gmail.com

Abstract

This paper describes our system (Fermi) for Task 4: Hyper-partisan News detection of SemEval-2019. We use simple text classification algorithms by transforming the input features to a reduced feature set. We aim to find the right number of features useful for efficient classification and explore multiple training models to evaluate the performance of these text classification algorithms. Our team - **Fermi**'s model achieved an accuracy of 59.10% and an F1 score of 69.5% on the official test data set.

In this paper, we provide a detailed description of the approach as well as the results obtained in the task.

1 Introduction

Hyper-partisan refers to a person or a group's tendency to be extremely partisan or biased towards a person or a group and specifically towards a political person or a political party. With the tremendous increase in citizen-based journalism, where anyone can create a website and post his (biased) views, there is a new phenomenon called fake news and it's potential role in affecting the election results, and has the ability to modify and impact the public's perception towards various people, companies and political parties. These kind of 'news' are usually one-sided, inflammatory, emotional and mostly woven around untruths. Combined with the proliferation of social media platforms, these 'fake news' signals get amplified and may potentially mask the signal of the real news. The fake news phenomenon hype has caused irreparable loss to many politicians, companies and in some cases involved the death of fellow citizens.

While Social media platforms can be used for constructive ideas, a small group of people can propagate their notions including hatred or affinity towards or against an individual, or a group or a race to the entire world in a few seconds. This necessitates the need to come up with computational methods to identify hyper-partisan news in user generated content.

Using computational methods to identify hyper-partisan news has been gaining attention in recent years as evidenced in (Potthast et al., 2018).

2 Related Work

In this section, we briefly describe other work in this area.

Hyper-partisan news detection is a new area and to the best of the knowledge of the authors, not much work has been done in this area. However, a close and related task is that of fake news detection. (Pérez-Rosas et al., 2017) use linguistic features to distinguish between fake and legitimate news content. (Wang, 2017) collect a decade long manually labelled sor statements in various context from a political fact checking website and create fake news classifiers using surface level linguistic patterns. (Tschiatschek et al., 2018) leverage crowd signals for detecting fake news. (Long et al., 2017) tackles the problem of fake news through multi-perspective speaker profiles.

Papers published in the last two years include the surveys by (Zhou and Zafarani, 2018), (Zhou et al., 2019) and (Shu et al., 2017), the paper by (Kumar and Shah, 2018).

A shared task on Hyper-partisan News detection(Kiesel et al., 2019) was announced as part of the annual workshop SemEval 2019. The task was to find if the given news article text and classify if it follows a hyper-partisan argumentation, i.e., whether it exhibits blind, prejudiced, or unreasoning allegiance to one party, faction, cause, or person.

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 954–956
Minneapolis, Minnesota, USA, June 6–7, 2019. ©2019 Association for Computational Linguistics

3 Methodology and Data

The data collection methods used to compile the data set in Hyperpartisan news detection is described in (Kiesel et al., 2019). We tackle the problem of identifying a piece of news as hyperpartisan or not by formulating it as a text classification problem. We use bag of words representation to transform the individual documents into vectors. After the transformation, we reduce the number of dimensions by using chi-square feature selection technique. In this method, the chi-square statistics between every feature variable and the target variable are computed, and then the existence of a relationship between the variables and the target is calculated. If the target variable is independent of the feature variable, that feature variable is not useful for prediction. If the two are dependent, then that feature variable is very important. In text classification, the feature selection is the process of selecting a specific subset of the terms of the training set and using only them in the classification algorithm. The feature selection process takes place before the training of the classifier. We use Random Forest Classifier from scikit-learn[1] machine learning library to generate models on these reduced features. The number of estimators in all the experiments is 20. All other parameters are default.

Our results on the different number of important features have been mentioned and described in the results section.

We haven't used any external datasets to augment the data for training our models.

No of features	F1 (macro)	Accuracy
200	0.39	0.51
400	0.40	0.51
600	0.42	0.51
800	0.44	0.52
1000	**0.46**	**0.52**

Table 1: *Dev* set Accuracy and Macro-F1 scores on labels by publisher dataset.

Dataset	F1 (macro)	Accuracy
Labels-by-article	0.69	0.59
Labels-by-publisher	0.66	0.61

Table 2: *Test set* Accuracy and Macro-F1 scores.

[1]https://scikit-learn.org/

4 Results and Analysis

Table 2 shows the dev set macro-averaged F-1 and accuracy for different number of important features.

We notice that the best performance was bagged by the model which uses 1000 features with Random Forest. We submitted this best model for evaluation on the test data and Table 4 shows the results.

The potential applications of this work show how different number of important features affect the performance of the classification task.

5 Future Work

Due to some constraints on the TIRA[2] platform, we were unable to use state-of-the-art deep learning techniques for text classification, which gained immense popularity in the past few years. In the future, we would like to explore transfer learning and deep learning algorithms to create models for and evaluate their performance for this task.

References

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Srijan Kumar and Neil Shah. 2018. False information on web and social media: A survey. *arXiv preprint arXiv:1804.08559*.

Yunfei Long, Qin Lu, Rong Xiang, Minglei Li, and Chu-Ren Huang. 2017. Fake news detection through multi-perspective speaker profiles. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 252–256.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2017. Automatic detection of fake news. *arXiv preprint arXiv:1708.07104*.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 231–240. Association for Computational Linguistics.

[2]https://tira.io

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, 19(1):22–36.

Sebastian Tschiatschek, Adish Singla, Manuel Gomez Rodriguez, Arpit Merchant, and Andreas Krause. 2018. Fake news detection in social networks via crowd signals. In *Companion of the The Web Conference 2018 on The Web Conference 2018*, pages 517–524. International World Wide Web Conferences Steering Committee.

William Yang Wang. 2017. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*.

Xinyi Zhou and Reza Zafarani. 2018. Fake news: A survey of research, detection methods, and opportunities. *arXiv preprint arXiv:1812.00315*.

Xinyi Zhou, Reza Zafarani, Kai Shu, and Huan Liu. 2019. Fake news: Fundamental theories, detection strategies and challenges. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 836–837. ACM.

Harvey Mudd College at SemEval-2019 Task 4: The Carl Kolchak Hyperpartisan News Detector

Celena Chen cechen@hmc.edu
Celine Park cpark@hmc.edu
Jason Dwyer jdwyer@hmc.edu
Julie Medero jmedero@hmc.edu

Abstract

We use various natural processing and machine learning methods to perform the Hyperpartisan News Detection task. In particular, some of the features we look at are bag-of-words features , the title's length, number of capitalized words in the title, and the sentiment of the sentences and the title. By adding these features, we see improvements in our evaluation metrics compared to the baseline values. We find that sentiment analysis helps improve our evaluation metrics. We do not see a benefit from feature selection. Overall, our system achieves an accuracy of 0.739, finishing 18th out of 42 submissions to the task. From our work, it is evident that both title features and sentiment of articles are meaningful to the hyperpartisanship of news articles.

1 Introduction

In 2019, Task 4 of the SemEval Workshop asked participants to automatically identify hyperpartisan texts (Kiesel et al., 2019). Hyperpartisan news detection is the problem of building a classifier using natural language processing techniques in order to label news articles as either hyperpartisan or neutral in content bias. Hyperpartisan articles, in this case, can be defined as articles which are very polarized and extremely biased towards one political party. This task is quite relevant in today's political climate, with reports of "fake news" articles heavily influencing votes and people's support of some candidates running for government offices. The issue is especially egregious because these biased news articles are informing political opinions of people who believe them to be factual and impartial, with no easy way to remove or detect hyperpartisan news articles. This is also a non-trivial task, as hyperpartisan news is not extremely explicit in its bias, and even human readers do not always agree on which articles should be classified as hyperpartisan or what about those articles merits such a classification. There is no one unifying feature of hyperpartisan news, and even news publishers which do produce hyperpartisan news are not guaranteed to *only* publish hyperpartisan news. Therefore, each article must be evaluated for its degree of hyperpartisanship, along many different axes of measurements.

A functional and accurate hyperpartisan news detector would be useful for social media sites and other carriers of news to make sure that their news content is unbiased, and to be able to detect and perhaps remove or block sources of hyperpartisan news. Facebook, for example, has been under great public scrutiny due to the quantity and popularity of hyperpartisan news on its site. Hyperpartisan news detection could also be useful for researchers seeking to understand the scope and impact of hyperpartisan news on the 2016 presidential election and how it can continue to inform voters today and in future elections.

2 Previous Work

Hyperpartisan news detection has become a popular application of natural language processing due to its relevance in contemporary politics. Specifically, there has been research to hash out what features are prevalent in hyperpartisan articles. Buzzfeed conducted a manual analysis of nine different pages on Facebook: three that were mainstream news, three that were hyperpartisan left, and three hyperpartisan right (Silverman et al., 2018). They rated every post as mostly false, mixture of true and false, mostly true, or not factual, for posts like memes or jokes. They determined that hyperpartisan articles on both the left and the right side have more in common with each other than with articles in the mainstream, and detecting whether or not an article was hyperpartisan was easier than detecting the actual orientation of the bias (Potthast et al., 2017). Likewise, in our application, we build a hyperpartisan news detector which labels hyperpartisanship but not whether an article is left- or right-leaning.

Fake news, which hyperpartisan sites are more likely to produce, tends to have certain qualities about its titles that make them distinct (Horne and Adali, 2017). These qualities are longer titles, simpler, more readable vocabulary words, and multiple words in the title which are all capitals. We use these qualities to inform our feature extraction of the article title, extracting the length of titles, the average length of title words, and the number of words in all capitals and adding these features to our larger feature matrix.

Polarity indicates how positive or negative a text may be, or the direction of the bias, while subjectivity indicates how strongly the text represents an opinion versus an objective fact, or the magnitude of the bias (Liu, 2010). We hypothesize that hyperpartisan news articles will carry relatively strong observable opinionation in comparison to non-hyperpartisan articles, so we use the two metrics of subjectivity and polarity as an addition to the other features in our feature matrix.

In this vein, one past study used sentiment analysis on the comment sections of articles about the Trayvon Martin case. It determined that more well known commentators tended to have stronger sentiment in their comments (Ignatow et al., 2016), implying that sentiment is a useful metric for analyzing opinions on the World Wide Web. Another study that used sentiment analysis on social media data showed that sentiment analysis was a key technique for extracting features of an opinion, allowing the researchers to propose models for simulating and forecasting online opinions (Kaschesky et al., 2011). This research in particular was interesting, because it covers a similar area of interest as hyperpartisan news detection. That is, it examines the far-reaching effects of political opinions and their proliferation on the World Wide Web, and also uses similar natural language processing techniques to extract information about these opinions. This tells us that sentiment analysis is an important tool for computational analysis of political opinions.

3 Methodology

Our model was trained and tested on the pre-labeled dataset provided by the SemEval group and the basis of our approach was a bag-of-words model. In order to improve upon the bag-of-words model and integrate some known salient features of hyperpartisan news, we also included headline features as well as sentiment analysis scores of the articles.

3.1 Data Set

To train our model, we use training data provided by the SemEval 2019 Hyperpartisan News Detection task organizers (Kiesel et al., 2019). These data come in the form of news articles given in XML format. Each article was given with title and article body text, and had labels provided in a separate file to indicate whether they had been flagged as hyperpartisan.

This data came in two distinct training sets as provided by the task organizers. The larger of the two sets, with about 800,000 labeled articles, was labeled by publisher; that is, publishers were grouped by whether they were known to be hyperpartisan in general, and the corresponding label was applied to all articles by a given publisher. A smaller set, comprising around 650 articles, was entirely hand-labeled; that is, human readers determined on an article-by-article basis whether a given article should be labeled as hyperpartisan.

It should be noted here that the smaller set labels are more true to what is expected of this task. Specifically, it is more of interest to us whether we can detect hyperpartisanship of articles based on how humans would judge it. Though the labeling by publisher is useful for obtaining a larger data set, it introduces some error due to the possibility that hyperpartisan sources may sometimes publish non-hyperpartisan articles and vice versa. Despite the advantages of the hand-labeled data set, its small size makes it much less feasible as a training set, so we also made substantial use of the larger set as we were tuning our model.

The content of each article was pre-processed with the Python library `spaCy` to tokenize and sentence-segment the text (AI, 2016–).

3.2 Feature Extraction

We used a bag-of-words approach to use for our main set of features, using a vocabulary of common English words. Unknown words were ignored. We filtered out 100 stop words, and used a vocabulary of 30,000 words.

We also included features from the titles, as certain qualities about the titles in hyperpartisan articles may be different than mainstream articles. We included the number of words in the title, hypothesizing that long titles can often indicate the article is misleading, or very biased. We included the number of fully capitalized words, which can often indicate that the article is not mainstream. Finally, we added a feature for the average length of the words in the title, as some research shows that hyperpartisan or biased articles tend to use more short, easily understood, words to appeal to the average reader (Horne and Adali, 2017).

We used the Python library TextBlob to do sentiment analysis on the articles and their titles (Loria, 2018). TextBlob has a sentiment analysis tool that provides both the subjectivity and polarity of a given sentence. We found the average subjectivity and average polarity of all the sentences in the article and used it as a feature. We also used the sentiment subjectivity and polarity of the article's title as a feature.

As our vocabulary used for extracting bag-of-words features was quite large, we used the built-in `SelectKBest` feature selection class provided by scikit-learn (Pedregosa et al., 2011) to narrow down the set of features we were using. However, testing with adjusting the parameters for feature selection did not seem to yield better results than simply using all possible features, so our final system made used of all of the available features.

3.3 Classifier

We feed our features to a multinomial naive Bayes (NB) classifier in scikit-learn (Pedregosa et al., 2011). For comparison to a baseline, we also use a majority-class dummy classifier.

4 Results

Table 1 shows the results of training with 10-fold cross-validation for each of our classifiers. As expected, the dummy classifier did not perform well. It represents a majority class baseline, though, and is useful for comparison purposes.

Our evaluation metrics improved with adding sentiment analysis and features of the title. Our final model does better than all of our previous models for every metric, including the dummy classifier, multinomial naive Bayes on BoW features, and adding title features. With an F-measure of 0.800, our precision and recall are well-balanced, and both are around 80%.

On the hand labeled SemEval test set, we achieved an accuracy of 0.739 and an F1 score of 0.745. Overall, our system ranked 18th out of 42 by accuracy, and 11th by F1 measure.

5 Discussion

We can infer from the results of our system that the features we extracted from the text, such as title features and sentiment, were significant and correlated to the hyperpartisanship of articles. This was expected, as the design of our classifier was based on previous work which determined that such features were useful for detecting bias in text. Since we combined different aspects of other studies, we were able to build upon previous findings and gain a more holistic view of hyperpartisan news articles Since this is the first offering of the SemEval Hyperpartisan News Detection task, we see our work as providing a foundation for future groups to build on as they attempt to fine-tune and improve a classifier for this important task.

There are still many questions regarding hyperpartisan news identification that remain unanswered. For example, it would be interesting to incorporate bigrams or trigrams of words instead of just using the bag-of-words approach. We also hypothesize that noun phrase chunks would be indicative of hyperpartisanship due to the ubiquity of certain controversial noun phrases in current media. The temporal nature of these features presents a unique challenge, though,

We could also use sentiment analysis in other ways. For instance, instead of just taking the average subjectivity and average polarity, it might also be interesting to find the percentage of sentences with a absolute value of polarity above a certain threshold. This could indicate an article is hyperpartisan if there are a lot of sentences that are above some threshold for polarity, that is, very opinionated sentences either strongly positive or strongly negative.

It would also be interesting to be able to look into the comment sections of the articles and determine if the sentiment of the comments can indicate hyperpartisanship. It seems probable that hyperpartisan articles would tend to attract more hyperpartisan viewers than mainstream articles would, and these people would have similarly strong opinions and be willing to voice them. The alignment of the comments may not even be aligned with the article, as the article may attract people from the other side, looking to critique or complain about the article. This data was not available for the SemEval task, but polarity and subjectivity also seem

Classifier	Accuracy	Precision	Recall	F-measure
DC: most frequent	0.553	0.553	1.0	0.712
NB (BoW only)	0.552	0.61	0.297	0.401
+title features	0.556	0.615	0.315	0.417
+sentiment features	0.793	0.781	0.820	0.800

Table 1: Cross-validation results for dummy classifier and a Naive Bayes classifier using bag of words features with and without additional features related to article title and sentiment.

like they would be useful metrics to extract from article comments.

In addition to the comments, it would be interesting to analyze how often the articles were shared, viewed, commented on, or in other ways interacted with. As the Buzzfeed study showed, hyperpartisan articles and articles that may not be entirely true tended to get more shares than non-partisan, as these are more interesting and inflammatory, and so this may be another feature that would have helped determine the hyperpartisanship of the article (Silverman et al., 2018).

Trying different classifiers would also be an appropriate next step. We focused on feature selection above experimenting with different classifiers because we believed that feature selection would give more meaningful insights into the nature of hyperpartisan articles than merely optimizing a classifier, but both are likely necessary to successfully identifying hyperpartisan articles.

6 Namesake

Our system is named after Carl Kolchak, the main character from the television series *Kolchak: The Night Stalker*, which aired in 1974-75. On the show, Kolchak investigated mysterious cases that had been abandoned by the police. We believe the unlikely and often unbelievable scenarios encountered by Kolchak would have been likely fodder for fake and hyperpartisan news during its time, and hope that our system will contribute to a community effort to automatically separate truth from fiction (Wikipedia contributors, 2019).

Figure 1: Darren McGavin, who portrayed Carl Kolchak in the TV Series *Kolchak: The Night Stalker* (ClassicBecky, 2011).

References

Explosion AI. 2016–. spacy: Industrial-strength natural language processing.

ClassicBecky. 2011. Kolchak: The night stalker ... ”the ripper”. Accessed: 2019-02-20.

Benjamin D. Horne and Sibel Adali. 2017. This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. *CoRR*, abs/1703.09398.

Gabe Ignatow, Nicholas Evangelopoulos, and Konstantinos Zougris. 2016. *Sentiment Analysis of Polarizing Topics in Social Media: News Site Readers Comments on the Trayvon Martin Controversy*, chapter 10. Emerald Group Publishing Limited.

Michael Kaschesky, Pawel Sobkowicz, and Guillaume Bouchard. 2011. Opinion mining in social media: Modeling, simulating, and visualizing political opinion formation in the web. In *Proceedings of the 12th Annual International Digital Government Research Conference: Digital Government Innovation in Challenging Times*, dg.o ’11, pages 317–326, New York, NY, USA. ACM.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Bing Liu. 2010. Sentiment analysis and subjectivity. In *Handbook of Natural Language Processing, Second Edition. Taylor and Francis Group, Boca*.

Steven Loria. 2018. Textblob: Simplified text processsing.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2017. A stylometric inquiry into hyperpartisan and fake news. *CoRR*, abs/1702.05638.

C. Silverman, L. Strapagiel, Shaban H., E. Hall, and J. Singer-Vine. 2018. Hyperpartisan facebook pages are publishing false and misleading information at an alarming rate. https://www.buzzfeednews.com/article/craigsilverman/partisan-fb-pages-analysis. Accessed: 2018-12-22.

Wikipedia contributors. 2019. Darren mcgavin — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Darren_McGavin&oldid=883055125. [Online; accessed 19-February-2019].

Harvey Mudd College at SemEval-2019 Task 4: The Clint Buchanan Hyperpartisan News Detector

Mehdi Drissi, Pedro Sandoval, Vivaswat Ojha, and Julie Medero
Harvey Mudd College, CA
{mdrissi, psandovalsegura, vmojha, jmedero}@hmc.edu

Abstract

We investigate the recently developed Bidirectional Encoder Representations from Transformers (BERT) model (Devlin et al., 2018) for the hyperpartisan news detection task. Using a subset of hand-labeled articles from SemEval as a validation set, we test the performance of different parameters for BERT models. We find that accuracy from two different BERT models using different proportions of the articles is consistently high, with our best-performing model on the validation set achieving 85% accuracy and the best-performing model on the test set achieving 77%. We further determined that our model exhibits strong consistency, labeling independent slices of the same article identically. Finally, we find that randomizing the order of word pieces dramatically reduces validation accuracy (to approximately 60%), but that shuffling groups of four or more word pieces maintains an accuracy of about 80%, indicating the model mainly gains value from local context.

1 Introduction

SemEval Task 4 (Kiesel et al., 2019) tasked participating teams with identifying news articles that are misleading to their readers, a phenomenon often associated with "fake news" distributed by partisan sources (Potthast et al., 2017).

We approach the problem through transfer learning to fine-tune a model for the document classification task. We use the BERT model based on the implementation of the github repository *pytorch-pretrained-bert*[1] on some of the data provided by Task 4 of SemEval. BERT has been used to learn useful representations for a variety of natural language tasks, achieving state of the art performance in these tasks after being fine-tuned (Devlin et al., 2018). It is a language representation model that is designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers. Thus, it may be able to adequately account for complex characteristics as such blind, prejudiced reasoning and extreme bias that are important to reliably identifying hyperpartisanship in articles.

We show that BERT performs well on hyperpartisan sentiment classification. We use unsupervised learning on the set of 600,000 source-labeled articles provided as part of the task, then train using supervised learning for the 645 hand-labeled articles. We believe that learning on source-labeled articles would bias our model to learn the partisanship of a source, instead of the article. Additionally, the accuracy of the model on validation data labeled by article differs heavily when the articles are labeled by publisher. Thus, we decided to use a small subset of the hand-labeled articles as our validation set for all of our experiments. As the articles are too large for the model to be trained on the full text each time, we consider the number of word-pieces that the model uses from each article a hyperparameter.

A second major issue we explore is what information the model is using to make decisions. This is particularly important for BERT because neural models are often viewed like black boxes. This view is problematic for a task like hyperpartisan news detection where users may reasonably want explanations as to why an article was flagged. We specifically explore how much of the article is needed by the model, how consistent the model behaves on an article, and whether the model focuses on individual words and phrases or if it uses more global understanding. We find that the model only needs a short amount of context (100 word pieces), is very consistent throughout an article, and most of the model's accuracy arises from locally examining the article.

In this paper, we demonstrate the effectiveness

[1] https://github.com/huggingface/pytorch-pretrained-BERT

of BERT models for the hyperpartisan news classification task, with validation accuracy as high as 85% and test accuracy as high as 77% [2]. We also make significant investigations into the importance of different factors relating to the articles and training in BERT's success. The remainder of this paper is organized as follows. Section 2 describes previous work on the BERT model and semi-supervised learning. Section 3 outlines our model, data, and experiments. Our results are presented in Section 4, with their ramifications discussed in Section 5. We close with an introduction to our system's namesake, fictional journalist Clint Buchanan, in Section 6.

2 Related Work

We build upon the Bidirectional Encoder Representations from Transformers (BERT) model. BERT is a deep bidirectional transformer that has been successfully tuned to a variety of tasks (Devlin et al., 2018). BERT functions as a language model over character sequences, with tokenization as described by Sennrich et al. (2016). The transformer architecture (Vaswani et al., 2017) is based upon relying on self-attention layers to encode a sequence. To allow the language model to be trained in a bidirectional manner instead of predicting tokens autoregressively, BERT was pretrained to fill in the blanks for a piece of text, also known as the Cloze task (Taylor, 1953).

Due to the small size of our training data, it was necessary to explore techniques from semi-supervised learning. Dai and Le (2015) found pre-training a model as a language model on a larger corpus to be beneficial for a variety of experiments. We also investigated the use of self-training (Zhu, 2005) to increase our effective training dataset size. Lastly, the motivation of examining the effective context of our classification model was based on Brendel and Bethge (2019). It was found that much higher performance than expected was achieved on the ImageNet dataset (Li Fei-Fei et al., 2009) by aggregating predictions from local patches. This revealed that typical ImageNet models could acquire most of their performance from local decisions.

[2]All of our code can be found here, https://github.com/hmc-cs159-fall2018/final-project-team-mvp-10000

3 Methodology

Next, we describe the variations of the BERT model used in our experiments, the data we used, and details of the setup of each of our experiments.

3.1 Model

We adjust the standard BERT model for the hyperpartisan news task, evaluating its performance both on a validation set we construct and on the test set provided by Task 4 at SemEval. The training of the model follows the methodology of the original BERT paper.

We choose to experiment with the use of the two different pre-trained versions of the BERT model, *BERT-LARGE* and *BERT-BASE*. The two differ in the number of layers and hidden sizes in the underlying model. *BERT-BASE* consists of 12 layers and 110 million parameters, while *BERT-LARGE* consists of 24 layers and 340 million parameters.

3.2 Training and Test Sets

We focus primarily on the smaller data set of 645 hand-labeled articles provided to task participants, both for training and for validation. We take the first 80% of this data set for our training set and the last 20% for the validation set. Since the test set is also hand-labeled we found that the 645 articles are much more representative of the final test set than the articles labeled by publisher. The model's performance on articles labeled by publisher was not much above chance level.

Due to an intrinsic limitation of the BERT model, we are unable to consider sequences of longer than 512 word pieces for classification problems. These word pieces refer to the byte-pair encoding that BERT relies on for tokenization. These can be actual words, but less common words may be split into subword pieces (Sennrich et al., 2016). The longest article in the training set contains around 6500 word pieces. To accommodate this model limitation, we work with truncated versions of the articles.

We use the additional $600,000$ training articles labeled by publisher as an unsupervised data set to further train the BERT model.

3.3 Experiments

We first investigate the impact of pre-training on *BERT-BASE*'s performance. We then compare the performance of *BERT-BASE* with *BERT-LARGE*. For both, we vary the number of word-pieces from each article that are used in training. We perform tests with 100, 250 and 500 word pieces.

We also explore whether and how the BERT models we use classify different parts of each individual article. Since the model can only consider a limited number of word pieces and not a full article, we test how the model judges different sections of the same article. Here, we are interested in the extent to which the same class will be assigned to each segment of an article. Finally, we test whether the model's behavior varies if we randomly shuffle word-pieces from the articles during training. Our goal in this experiment is to understand whether the model focuses on individual words and phrases or if it achieves more global understanding. We alter the the size of the chunks to be shuffled (N) in each iteration of this experiment, from shuffling individual word-pieces ($N = 1$) to shuffling larger multiword chunks.

4 Results

Our results are primarily based on a validation set we constructed using the last 20% of the hand-labeled articles. It is important to note that our validation set was fairly unbalanced. About 72% of articles were not hyperpartisan and this mainly arose because we were not provided with a balanced set of hand-labeled articles. The small validation split ended up increasing the imbalance in exchange for training on a more balanced set. The test accuracies we report were done on SemEval Task 4's balanced test dataset.

4.1 Importance of Pre-training

Our first experiment was checking the importance of pre-training. We pre-trained BERT-base on the 600,000 articles without labels by using the same Cloze task (Taylor, 1953) that BERT had originally used for pre-training. We then trained the model on sequence lengths of 100, 250 and 500. The accuracy for each sequence length after 100 epochs is shown in 1 and is labeled as UP (unsupervised pre-training). The other column shows how well *BERT-base* trained without pre-training. We found improvements for lower sequence lengths, but not at 500 word pieces. Since the longer chunk should have been more informative, and since our hand-labeled training set only contained 516 articles, this likely indicates that BERT experiences training difficulty when dealing with long sequences on such a small dataset. As the cost to do pre-training was only a one time cost all of our remaining experiments use a pre-trained model.

Max Seq Len	BERT-base	BERT-base + UP
100	76.7	79.8
250	75.9	82.9
500	79.1	75.2

Table 1: Validation accuracy for BERT-base with and without Unsupervised Pre-training (UP).

We evaluated this model on the *SemEval 2019 Task 4: Hyperpartisan News Detection* competition's **pan19-hyperpartisan-news-detection-by-article-test-dataset-2018-12-07** dataset using TIRA (Potthast et al., 2019). Our model, with a maximium sequence length of 250, had an accuracy of 77%. It had higher precision (83.2%) than recall (67.8%), for an overall F1-score of 0.747.

4.2 Importance of Sequence Length

Next, we further explore the impact of sequence length using *BERT-LARGE*. The model took approximately 3 days to pre-train when using 4 NVIDIA GeForce GTX 1080 Ti. On the same computer, fine tuning the model on the small training set took only about 35 minutes for sequence length 100. The model's training time scaled roughly linearly with sequence length. We did a grid search on sequence length and learning rate.

Table 2 shows that the model consistently performed best at a sequence length of 100. This is a discrepancy from *BERT-BASE* indicating that the larger model struggled more with training on a small amount of long sequences. For our best trained *BERT-LARGE*, we submitted the model for evaluation on TIRA. Surprisingly, the test performance (75.1%) of the larger model was worse than the base model. The experiments in (Devlin et al., 2018) consistently found improvements when using the large model. The main distinction here is a smaller training dataset than in their tasks. The experiments in the remaining sections use the same hyperparameters as the optimal *BERT-LARGE*.

4.3 Model Consistency

Due to the small training dataset, we tried self-training to increase our effective training set. We trained the model for 40 epochs. For the remaining 60 epochs, after each epoch we had the model make predictions on five slices of 500 unlabeled

Max Seq Len \ Learning Rate	5e-7	1e-6	1.5e-6	2e-6	2.5e-6	3e-6
50	78.3	80.6	79.8	79.1	79.1	77.5
100	83.7	83.7	86.1	86.1	85.3	84.5
150	77.5	79.8	81.4	80.6	79.8	79.8
200	81.4	80.6	79.8	84.5	83	81.4

Table 2: Validation Accuracy on *BERT-LARGE* across sequence length and learning rate.

articles. If an article had the same prediction for more than four slices, we added it to the labeled training data. The model always added every article to the training set, though, since it always made the same prediction for all 5 slices. This caused self-training to be ineffective, but also revealed that the model's predictions were very consistent across segments of a single article.[3]

4.4 Effective Model Context

Finally, we investigate whether the model's accuracy primarily arose from examining words or short phrases, or if the decisions were more global. We permuted the word pieces in the article at various levels of granularity. At the finest level (permute_ngrams = 1), we permuted every single word piece, forcing the model to process a bag of word pieces. At coarser levels, ngrams were permuted. As the sequence length for these experiments was 100, permute_ngrams = 100 corresponds to no permutation. The results can be found in 3.

permute_ngrams	Validation Accuracy
1	67.4
2	62.8
3	75.2
4	83.0
5	76.0
10	82.2
20	76.7
50	79.8
100	84.5

Table 3: *BERT-LARGE* across permute_ngrams.

Accuracy drops a lot with only a bag of word pieces, but still reaches 67.4%. Also, most of the accuracy of the model (within 2%) is achieved with only 4-grams of word pieces, so the model is not getting much of a boost from global content.

[3]We also tried training a model that averaged its predictions across multiple slices. This turned out to be slightly worse, likely due to the model's high consistency.

5 Discussion

Our successful results demonstrate the adaptability of the BERT model to different tasks. With a relatively small training set of articles, we were able to train models with high accuracy on both the validation set and the test set.

Our models classified different parts of a given article identically, demonstrating that the overall hyperpartisan aspects were similar across an article. In addition, the model had significantly lower accuracy when word pieces were shuffled around, but that accuracy was almost entirely restored when shuffling around chunks of four or more word pieces, suggesting that most of the important features can already be extracted at this level.

In future work, we we would like to make use of the entire article. Naively, running this over each chunk would be computationally infeasible, so it may be worth doing a full pass on a few chunks and cheaper computations on other chunks.

6 Namesake

Figure 1: Jerry verDorn as Clint Buchanan.

Our system is named after Clint Buchanan[4], a fictional journalist on the soap opera *One Life to Live*. Following the unbelievable stories of Clint and his associates may be one of the few tasks *more* difficult than identifying hyperpartisan news.

[4]http://abc.go.com/shows/one-life-to-live/bio/clint-buchanan/165745

References

Wieland Brendel and Matthias Bethge. 2019. Approximating CNNs with bag-of-local-features models works surprisingly well on imagenet. In *International Conference on Learning Representations*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3079–3087. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv e-prints*.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Li Fei-Fei, Wei Dong, Jia Deng, Kai Li, R. Socher, and Li-Jia Li. 2009. ImageNet: A large-scale hierarchical image database. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. Tira integrated research architecture. *In Nicola Ferro and Carol Peters, editors, Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF. Springer.*

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2017. A stylometric inquiry into hyperpartisan and fake news. *CoRR*, abs/1702.05638.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725. Association for Computational Linguistics.

Wilson L Taylor. 1953. cloze procedure: A new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Xiaojin Jerry Zhu. 2005. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Harvey Mudd College at SemEval-2019 Task 4: The D.X. Beaumont Hyperpartisan News Detector

Evan Amason
Harvey Mudd College
301 Platt Boulevard
Claremont, CA 91711
eamason@hmc.edu

Jake Palanker
Harvey Mudd College
301 Platt Boulevard
Claremont, CA 91711
jpalanker@hmc.edu

Mary Clare Shen
Harvey Mudd College
301 Platt Boulevard
Claremont, CA 91711
mshen@hmc.edu

Julie Medero
Harvey Mudd College
301 Platt Boulevard
Claremont, CA 91711
jmedero@hmc.edu

Abstract

We use the 600 hand-labelled articles from SemEval Task 4 (Kiesel et al., 2019) to hand-tune a classifier with 3000 features for the Hyperpartisan News Detection task. Our final system uses features based on bag-of-words (BoW), analysis of the article title, language complexity, and simple sentiment analysis in a naive Bayes classifier. We trained our final system on the 600,000 articles labelled by publisher. Our final system has an accuracy of 0.653 on the hand-labeled test set. The most effective features are the Automated Readability Index and the presence of certain words in the title. This suggests that hyperpartisan writing uses a distinct writing style, especially in the title.

1 Introduction

Hyperpartisan news is becoming more mainstream as online sources gain popularity. Hyperpartisan news is news written from an extremely partisan perspective, such that the goal is reinforcing existing belief structures in the party's ideology rather than conveying facts. Such hyperpartisan writing tends to amplify political divisions and increase animosity between opposing political ideologies. Hyperpartisan news sources also output fake news at startling rates (Silverman et al., 2016). Automatic detection of fake news is difficult, but detecting hyperpartisan news can help, and it can also expose biases in journalism. This task is challenging to automate because it is even difficult for humans: fake and biased news articles get shared on social media at high rates, and even labels that were hand-generated by professionals have errors (Silverman et al., 2016). We attempt to use various features of political news articles to train a multinomial Naive Bayes classifier to complete this task. We use a set of bag-of-words (BoW) features for words appearing in the title of each article, and for words appearing in the article text. With these features, we identified a set of words that characterize hyperpartisan writing. We also considered complexity features such as type-to-token ratio and automated readability index. Based on the performance of these features we attempt to answer the question of whether hyperpartisan writing is more or less complex than non-hyperpartisan writing. A successful classifier could be very useful in today's society. For example, it could be used to create a browser plug-in to check online articles for political bias in real time as the user reads. People on social media could use it to verify the legitimacy of a political article before sharing it with their followers. Encouraging people to share factual news rather than inflammatory hyperpartisan articles would hopefully improve communication between opposing parties and create a more informed population.

The rest of this paper begins with a description of previous work on the related task of fake news detection in Section 2. We then describe our model and features in Section 3, and our results in Section 4. Section 5 discusses some lessons learned with respect to what features are most useful in identifying hyperpartisan news, and Section 6 closes with a brief description of our system's namesake, fictional magazine editor D.X. Beaumont.

2 Previous Work

Since the 2016 election, there has been a lot of interest in fake news, which is closely related to the hyperpartisan news we focus on. Our approach to the hyperpartisan news task leverages lessons learned in prior work on fake news detection, and explores the extent to which that work is successful in a different but related task. Fake news detection has been widely studied (e.g., the survey paper by Fuhr et al. (Fuhr et al., 2018)), and we base many of our classifier's features on previous studies of fake news.

The content of fake and real news articles differ substantially. Fake news articles have been found to require a lower reading level than real news articles, to be less technical, and to use more personal pronouns. Further, their titles tend to be longer, use more proper nouns, and use more words that are all capitalized (Horne and Adali, 2017). Our work differs in that we were trying to determine whether an article is hyperpartisan, which is similar to but not the same as identifying fake news articles. In particular, a hyperpartisan news article may be factually correct (i.e., not contain any mistruths) but still be written with a hyperpartisan slant. We hypothesize, nonetheless, that the stylistic features that distinguish between real and fake news may be useful in identifying hyperpartisan news articles. Potthast, et. al., also showed that there are significant stylistic differences between hyperpartisan and mainstream news articles (Potthast et al., 2017). Consequently, we include reading level and features of each article's title as features in our model.

The success of these features on identifying fake news motivates our decision to focus on article titles as a differentiating feature, and to include reading level in the set of features available to our model.

Perez-Rosa et al. also examine fake news articles to create a classifier for them (Prez-Rosas et al., 2018). Their results identify additional features related to text readability, with fake news articles tending to be written at a lower reading level than real news articles. We incorporate features from their work, including *Average Word Length*, *Type-Token-Ratio*, and *SMOG Readability Formula* .

3 Methodology

Each article's content and title was tokenized using `spacy`'s default English model (AI, 2016–).

We use a multinomial naive Bayes classifier from `scikit-learn`, extracting a large number of features and then using feature selection to reduce the number of features available to our classifier.

3.1 Features

We make use of features related to the words in the article as a whole, the title of the article, sentiment, and text complexity.

Bag of Words Features: Using a vocabulary of 30,000 words, we count the number of times each vocabulary word occurs in the full article text. We then drop a fixed number stop words, selected automatically by frequency. We experimented with both 50 and 100 stop words, and the run of our system that was submitted to the SemEval task used 50 stop words.

Title Bag of Words: Next, using the same vocabulary but without excluding stop words, we add word counts for the title of the article. We also count the number of words in the title that are entirely capitalized, generally a feature of hyperpartisan titles (Horne and Adali, 2017).

Sentiment Analyzer: We use two sentiment lexicons (Hu and Liu, 2004). The first contains 2000 words with positive sentiment, and the second contains 4000 words with negative sentiment. We count the occurrence of words from each list, hypothesizing that hyperpartisan articles will likely have many more words with polarized sentiment than non-hyperpartisan articles.

Complexity Features: Finally, we include features designed to capture the articles' complexity. This category includes features such as *Average Word Length*, *Type-Token-Ratio*, and *SMOG Readability Formula*. Each of these is designed to capture the complexity of a given text; *Average Word Length* gives us insights into the vocab choices and uses of ”advanced” words, *Type-Token-Ratio* measures the amount of ”novel” words in the text, the *SMOG Readability Formula* is based on the number of polysyllabic words per sentence (which is influenced both by vocabulary choice, and sentence length). Since prior work shows that hyper-partisan articles are often written at an easier reading level, with more repeating words, and simpler sentence structure, we expect that these complexity features will be useful in identifying hyperpartisan articles.

3.2 Feature Selection

The above feature space was very large compared to the number of available articles, so we implemented two different methods of feature selection: one using variance, and one using a χ^2 test. In each case, we perform statistics on the training set, attempting to describe which features are the most *distinguishing*. Given these statistics, we score each feature, and select a subset of the total feature set using either a threshold score or a target feature count. By experimenting on the smaller hand-labeled data set, we found that reducing to the best 3000 features maximized our performance for 10-fold cross validation. This modification was made after the evaluation, however; our results on the SemEval task represent the performance of our task without feature selection.

4 Results

Our final system achieved an accuracy of 0.653, which ranked 28th out of 42 submissions on the test set hand-labeled by article.

4.1 Feature Selection

As part of additional analysis, we examined the effectiveness of feature selection on the validation set. Table 1 shows that reducing the number of features to 3000 had a negligible effect on both accuracy and f-measure. Since the validation set is qualitatively different from the hand-labeled test set used in the official competition, these results are not directly comparable to our final system performance. In particular, we note that our system performs slightly better on the validation set than on the test set regardless of the number of features used, which may indicate that our classifier learned some characteristics of the source-labeled validation set that distinguished it from the hand-labeled test set.

Feature Selection	Accuracy	f1-measure
`all`	0.611	0.675
`3000`	0.5983	0.667

Table 1: Validation set performance using `all` of our features or the `3000` most informative features.

5 Discussion

Hyperpartisan news has been a concern since the rise of social media, and that concern has only grown since the 2016 election. Giving consumers of social media the knowledge of whether or not what they are reading is hyperpartisan could help to reduce the number of people fooled by fake or misleading facts, and it could help to reduce the partisan divide within the United States.

Feature Title	Category	χ^2	p-value
”trump”	Polarity	416	1.77e-92
A.R.I.	Complexity	377	3.67e-84
”*”	Title	208	2.95e-47
”class”	Title	179	9.45e-41
”american”	Title	170	7.41e-39
”most”	Title	143	5.04e-33
”political”	Title	137	1.08e-31
”israel”	Title	133	1.16e-30
”like”	Polarity	128.7	7.85e-30
”these”	Title	126	2.92e-29

Table 2: Highest ranked features from our hand labeled data-set.

Using our χ^2 feature selection system, we found the top 10 features over the hand-labeled article set, shown in Table 2. The size of the hand-labeled set is rather small, so the extremely small p-values are likely inflated by this.

The Automated Readability Index feature (a complexity feature measuring word length and sentence length) is the second highest performing, indicating that this way of capturing complexity is worthy of further study.

A number of BoW features on the title are also important. The selected words included fall under a few categories such as controversial topic (*trump*, *Israel*), generalization (*most*, *these*), and political terms (*political*, *class*). Some, like the presence of ”*” in title, seem like strange outliers that are likely a consequence of a combination of formatting artifacts and the small size of the hand-labeled dataset.

While an earlier, simpler version of our model achieved 10-fold cross-validation accuracy of .787 on the hand-labeled training set, the submission we submitted performed much more poorly on the final test set. We hypothesize that one source of this difference may have been in the tuning of our hyper-parameter related to feature selection. We tuned this parameter manually using results from 10-fold cross-validation on the hand labeled data-set. Because the hand labeled data was significantly smaller, it is possible that it took far fewer features to properly classify the space. Improved tuning of this parameter on a larger set could have given us better results. Nonetheless, our work demonstrates that BoW, complexity, and polarity features are all useful in identifying hyperpartisan news articles.

6 Namesake

Our system is named after D.X. Beaumont, a magazine editor and publisher on the short-lived TV Series *My Sister Eileen* that aired on CBS in 1960-61 (Wikipedia contributors, 2018). The series, based on autobiographical short stories published in The New Yorker by Ruth McKenney (Lippman, 2018). Ruth, who aspired to be a writer, worked for Beaumont (shown in Figure 1 as portrayed by Raymond Bailey). We imagine that the proliferation of hyperpartisan news in modern communication would have caused the orderly Ruth a great deal of frustration, and hope that our contribution to this task will benefit future writers and their publishers.

Figure 1: Darren McGavin, who portrayed D.X. Beaumont in the TV Series *My Sister Eileen*(NBC Television, 2017).

Acknowledgments

We would like to thank our stalwart grutor (a Harvey Mudd portmanteau of *grader* and *tutor*!), Jonah Rubin, for his help at all hours on our coursework during the semester that led to this system submission.

References

Explosion AI. 2016–. spacy: Industrial-strength natural language processing.

Norbert Fuhr, Anastasia Giachanou, Gregory Grefenstette, Iryna Gurevych, Andreas Hanselowski, Kalervo Jarvelin, Rosie Jones, YiquN Liu, Josiane Mothe, Wolfgang Nejdl, Isabella Peters, and Benno Stein. 2018. An information nutritional label for online documents. *SIGIR Forum*, 51(3):46–66.

Benjamin D. Horne and Sibel Adali. 2017. This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. In *The 2nd International Workshop on News and Public Opinion at ICWSM*. Cornell University.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *KDD*.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Laura Lippman. 2018. In praise of ruth mckenney. *The New York Times*. [Online; accessed 18-February-2019].

NBC Television. 2017. Accessed: 2019-02-20 (Public domain). [link].

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2017. A stylometric inquiry into hyperpartisan and fake news. *CoRR*, abs/1702.05638.

Vernica Prez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic detection of fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*.

Craig Silverman, Lauren Strapagiel, Hamza Shaban, Ellie Hall, and Jeremy Singer-Vine. 2016. Hyperpartisan facebook pages are publishing false and misleading information at an alarming rate.

Wikipedia contributors. 2018. My sister eileen — Wikipedia, the free encyclopedia. [Online; accessed 18-February-2019].

NLP@UIT at SemEval-2019 Task 4: The Paparazzo Hyperpartisan News Detector

Duc-Vu Nguyen$^\diamondsuit$**, Dang Van Thin**$^\diamondsuit$**, and Ngan Luu-Thuy Nguyen**$^\heartsuit$
$^\diamondsuit$Multimedia Communications Laboratory
$^\heartsuit$Faculty of Computer Science
University of Information Technology, Vietnam National University Ho Chi Minh City, Vietnam
{vund,thindv,ngannlt}@uit.edu.vn

Abstract

This paper describes the system of NLP@UIT that participated in Task 4 of SemEval-2019. We developed a system that predicts whether an English news article follows a hyperpartisan argumentation. Paparazzo is the name of our system and is also the code name of our team in Task 4 of SemEval-2019. The Paparazzo system, in which we use *tri*-grams of words and *hepta*-grams of characters, officially ranks thirteen with an accuracy of 0.747. Another system of ours, which utilizes *tri*-grams of words, *tri*-grams of characters, *tri*-grams of part-of-speech, syntactic dependency sub-trees, and named-entity recognition tags, achieved an accuracy of 0.787 and is proposed after the deadline of Task 4.

1 Introduction

Fake news is a noteworthy term in recent years. The rise of users and rapid spread information on social networking have made on automatic controlling of fake news more difficult. Fake news articles are typically extremely one-sided (hyperpartisan), inflammatory, emotional, and often riddled with untruths (Potthast et al., 2018). The influence of misinformation varies depending on the style it is written in. For example, sarcasm in a sports news article will have less of an impact than news written in the hyper-partisan argumentation style, which can sway voter decision in an election.

Hyperpartisan detection in news articles is one of the ways to control fake news on the media and public. Kiesel et al. (2019) provided a new task, which they name "Hyperpartisan News Detection," to decide whether a news article text follows a hyperpartisan argumentation. We approach this task following traditional text classification by extracting style features. The bag-of-words model is the way of text representation and is applied to sentiment analysis effectively (Pang et al., 2002). Matsumoto et al. (2005) applied text mining techniques on dependency sub-trees as features for sentiment analysis at the document level. Our results show that n-grams of words and dependency sub-trees features from sentences of the document have certain impacts on the performance of the classifier. The details of the features in our systems and the results are described in Section 3 and Section 4.

2 Task Description

SemEval2019 Task 4 has only one task, in which participants are required to build the systems for hyperpartisan news detection. The task is to predict which category ("hyper-partisan" vs. "not hyper-partisan") an argumentation belongs to when given the news article in English (Kiesel et al., 2019). There are 645 articles in the for-ranking training set, and 628 articles in the for-ranking testing set (all of them are labeled through crowdsourcing on an article basis). Besides, the organizers of this task provided another dataset with the training/validation/testing set having 600,000/150,000/4,000 articles (all of them are labeled in accordance with the judgment of the publisher). The organizers use the *accuracy* as the main metric in the *for-ranking testing set* to evaluate the performance of the participants' systems. All submissions and results are validated by the organizers via the evaluation service TIRA (Potthast et al., 2019).

3 System Description

In this section, we describe the major stages we followed, as well as the prediction models we utilized in our detection system.

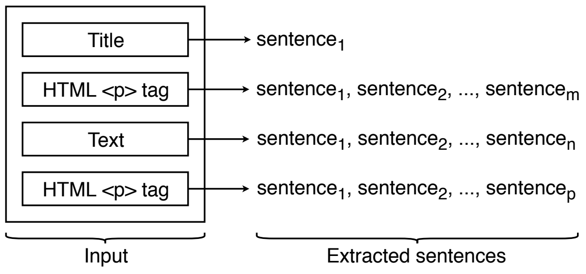

Title
HTML <p> tag
Text
HTML <p> tag
sentence$_1$
sentence$_1$, sentence$_2$, ..., sentence$_m$
sentence$_1$, sentence$_2$, ..., sentence$_n$
sentence$_1$, sentence$_2$, ..., sentence$_p$
Input
Extracted sentences

Figure 1: Diagram of data preprocessing.

3.1 Data Preprocessing

Data preprocessing of the given input is the important phase for every task related to natural language processing. The input of SemEval-2019 Task 4 is an XML file, containing a title and many paragraphs in the body text. Paragraph segmentation is based on the HTML <p> tag because the <p> tag defines a paragraph. While many paragraphs are wrapped by the <p> tag, some are not. Observation of some inputs from the dataset shows that paragraphs that are not wrapped by any HTML tag may contain "noise," such as advertisements and the browser's error messages. On the other hand, texts displayed in HTML <p> tags can also contain "noise," such as notifications for redirecting a page (e.g., "Click here to..."). We did not handle the aforementioned noises in our experiment.

The next step after paragraph segmentation is sentence segmentation. During this process, we used spaCy tool (Honnibal and Montani, 2017) to extract sentences from titles, HTML <p> tags, and paragraphs not wrapped in any HTML tag of input (as we can see the diagram in Figure 1).

3.2 Features Extraction

3.2.1 N-grams of words

Before extracting n-grams of words, we break the sentences into words in three ways:

1. WS$_1$: The sentence is split by space/multi-space into tokens.
2. WS$_2$: The sentence is split by space/multi-space into tokens. After that, we discard tokens which are punctuations or English stop-words.
3. WS$_3$: The sentence is segmented into words. And then, we lemmatize words into lemmas. All is done by using the spaCy tool (Honnibal and Montani, 2017).

After splitting/segmenting the sentence into tokens/words, we put tokens/words are all in lowercase and implement extracting n-grams of them. The specific values of n for prediction models are mentioned in section 3.3.

3.2.2 N-grams of characters

Extracting n-grams of words is effective for text classification that is word-based representation, but this approach requires reliable tokenizers for breaking the sentences into words. Experiments on unsolicited e-mail messages (spam) and a variety of evaluation measures, Kanaris et al. (2007) show that n-grams of characters are more reliable to classify texts than n-grams of words. Potthast et al. (2018) show how a style analysis can distinguish hyperpartisan news from the mainstream, and they also use tri-grams of characters as features for the classifier in their experiments. As we described in section 3.1, unfortunately, the input of SemEval-2019 Task 4 contains a small number of strange n-grams of characters towards the tokenizers. Therefore, we decide to use n-grams of characters as the features in our system. We use the sentence with all of its tokens being rejoined after the segmentation in WS$_1$ (we described in section 3.2.1) with character space for extracting n-grams of characters. In our experiments, the value of n ranging from 2 to 7 and the specific values for prediction models are mentioned in section 3.3.

3.2.3 N-grams of part-of-speech

Argamon et al. (2003) found that n-grams of part-of-speech can efficiently capture syntactical information and gender-based style of the writer. Potthast et al. (2018) used tri-grams of part-of-speech to make a comparative style analysis of hyperpartisan (extremely one-sided) news and fake news. Although the efficacy of using n-grams of part-of-speech on fake news was not examined in their study, we decided to experiment by using n-grams of part-of-speech as features for hyper-partisan news detection. We used the spaCy tool (Honnibal and Montani, 2017) for part-of-speech tagging and extract tri-grams of part-of-speech as features.

3.2.4 Sub-trees of dependency tree

In our experiment, dependency parsing involves extracting from a dependency tree a dependency sub-tree, which is defined by Matsumoto et al. (2005) as "a tree obtained by removing zero or more nodes and branches from the original de-

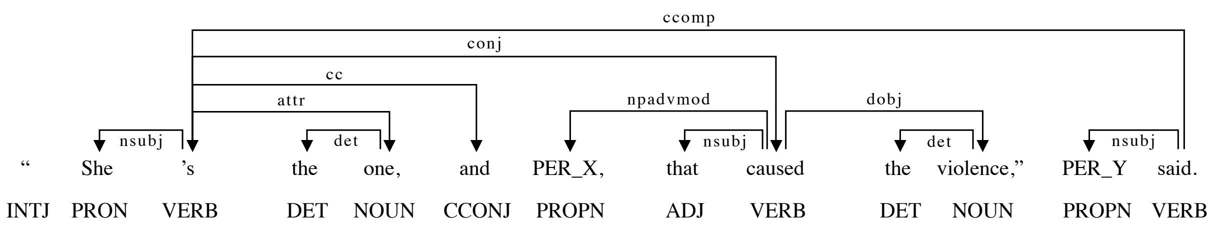

ccomp
conj
cc
attr
npadvmod
dobj
nsubj
det
nsubj
det
nsubj
“ She ’s the one, and PER_X, that caused the violence,” PER_Y said.
INTJ PRON VERB DET NOUN CCONJ PROPN ADJ VERB DET NOUN PROPN VERB

Figure 2: Visualization of the dependency tree of the sentence within the bracket (“She’s the one, and PER_X, that caused the violence,” PER_Y said.). This sentence is taken from a news article of the training for-ranking training set which is mentioned in Section 2. The person’s name is replaced by PER_{uppercase letter} in this example (we did not do that in our experiment).

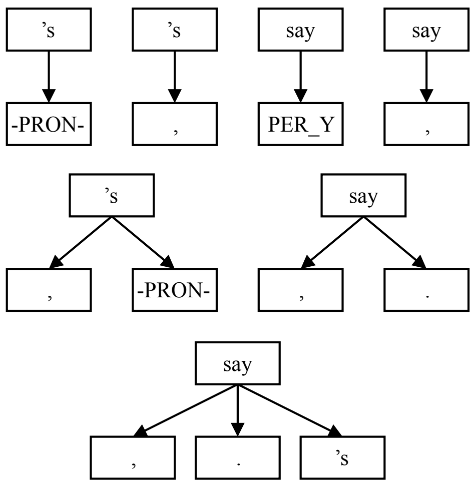


Figure 3: Visualization of seven sub-trees which are extracted from the dependency tree in Figure 2. There are four sub-trees with two nodes, two sub-trees with three nodes, one sub-tree with four nodes in the current figure. All words in this example are lemmatized.

pendency tree.” Figure 2 illustrates a dependency tree of a sentence parsed with spaCy tool (Honnibal and Montani, 2017), and its shortcoming that shows the double quotation mark on the left does not have any child node or parent node. This shortcoming, however, did not affect the extraction of all sub-trees of the dependency tree, but we resolved this issue by considering each group of sub-trees as one connected component, and the dependency tree as a graph that can contain more than one connected component. Figure 3, the number of nodes in a sub-tree can range from 2 to 4, and NetworkX tool (developed by Hagberg et al. (2008)) was used to extract all the sub-trees of the original dependency tree as one connected component for each node. All words at each node of sub-trees are lemmatized in our experiment. As we can see in Figure 3, some sub-trees can capture words which are not located close to each other.

3.2.5 Named-entity recognition tags

Characteristics of the input of SemEval-2019 Task 4 contains names of people, names of organizations. Therefore, we decided to use mentions of specific terms in named-entity recognition as features. In our experiments, a feature is represented by concatenating a mention and a named-entity recognition tag. We used the spaCy tool (Honnibal and Montani, 2017) for the named-entity recognition task.

3.3 Prediction Models

In this section, we describe the four models which we have summited to the organizers. In all models, we used linear SVM (SGDClassifier from Scikit-learn (Pedregosa et al., 2011)) as the classifier, and the loss function which is hinge loss with L2 regularization. In all models, we did not run validation experiments for turning regularization term α of all models. We used just the default value of $\alpha = 0.0001$ following SGDClassifier from Scikit-learn (Pedregosa et al., 2011). Most importantly, we concatenated different count vectors by way of extracting features described in section 3.2, to obtain the input representation of the model.

3.3.1 First model

This model uses *tri*-grams of words which are split from the text of the article (we did not segment the text into sentences) the way described in WS_1 in section 3.2.1). Besides, the first model uses *hepta*-grams of characters from the text of the article as features. We discarded the title when extracting the features for the first model, and we do not distinguish between texts with the HTML <p> tag wrapping and those without (as mentioned in section 3.1).

3.3.2 Second model

We extracted bi-grams of characters from the body text regardless of whether the text is wrapped by the HTML <p> tag or not, and for the title, we followed the way mentioned in WS_2 in section 3.2.1) to extract its bi-grams of words. Additionally, we extracted all mentions of named-entity recognition from all sentences of the article, and we distinguished between features from the title and those from the body text.

3.3.3 Third model

This model shares similar features with the second one, except for our extraction of the dependency sub-trees.

3.3.4 Fourth model

In this model, the title, the text with the HTML <p> wrapping, and those without are distinguished. All sentences are segmented to tokens in the same way described in WS_3 in section 3.2.1). We used tri-grams of words, tri-grams of characters, tri-grams of part-of-speech, syntactic dependency sub-trees, and named-entity recognition tags to extract the text before performing the TF-IDF transformation with Scikit-learn tool (developed by Pedregosa et al. (2011)) on the combined features with min_df at 0.05 and max_df at 0.95.

4 Results

	Accuracy	Precision	Recall	F1
First model[1]	0.747	0.754	0.732	0.743
Second model	0.685	0.687	0.678	0.683
Third model	0.707	0.666	0.831	0.739
Fourth model[2]	0.787	0.796	0.771	0.783

Table 1: Metric summary of fully trained models on the official test dataset.

We did not use the training dataset of 600,000 articles for training all the models in our experiments. The result (Table 1) shows a decrease in performance of the second and the third models when the n-grams of words were not used as features. The accuracy of the third model, however, increased by 2% compared with the second model when the extra dependency sub-trees were used as features. On the other hand, the fourth model achieved the highest accuracy, up to 0.787.

[1]The first model officially ranks thirteen in Sem-Eval 2019 Task 4.

[2]The fourth model is proposed after the deadline of Sem-Eval 2019 Task 4.

This accuracy level, however, is still lower than that achieved via deep learning techniques, such as the Convolutional neural network and pre-trained ELMo representations, employed by "Bertha von Suttner" team who were ranked first in SemEval-2019 Task 4.

5 Conclusion

Our major contribution to SemEval-2019 Task 4 is that using n-grams of words and dependency sub-trees as features for extracting has a positive impact on the performance of the classifier: In our experiment, we were able to achieve the accuracy of 0.787 with the proposed model that uses tri-grams of words, tri-grams of characters, tri-grams of part-of-speech, syntactic dependency sub-trees, and named-entity recognition tags. That model can also capture words which are not located close to each other through dependency sub-trees. The disadvantages of our models, however, are that extraction of dependency sub-trees is a time-consuming process, and the relations between sentences of the articles are not represented.

References

Shlomo Argamon, Moshe Koppel, Jonathan Fine, and Anat Rachel Shimoni. 2003. Gender, genre, and writing style in formal written texts. *Text*, 23:321–346.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, pages 11 – 15.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Ioannis Kanaris, Konstantinos Kanaris, Ioannis Houvardas, and Efstathios Stamatatos. 2007. Words versus Character n-Grams for Anti-Spam Filtering. *International Journal on Artificial Intelligence Tools*, 16:1047–1067.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Shotaro Matsumoto, Hiroya Takamura, and Manabu Okumura. 2005. Sentiment Classification Using Word Sub-sequences and Dependency Sub-trees. In

Advances in Knowledge Discovery and Data Mining, pages 301–311, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 79–86. Association for Computational Linguistics.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. TIRA Integrated Research Architecture. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 231–240. Association for Computational Linguistics.

Orwellian-times at SemEval-2019 Task 4: A Stylistic and Content-based Classifier

Jürgen Knauth
Institute of Computer Science
University of Goettingen
jknauth@uni-goettingen.de

Abstract

While fake news detection received quite a bit of attention in recent years, hyperpartisan news detection is still an underresearched topic. This paper presents our work towards building a classification system for hyperpartisan news detection in the context of the SemEval2019 shared task 4. We experiment with two different approaches - a more stylistic one, and a more content related one - achieving average results.

1 Introduction

Recent years have seen a noticeable change in the political discourse: Political polarization has increased and political opinions have become more hyperpartisan (Doherty, 2017). This affects the media, especially news media and is therefore a topic of considerable interest for science and society.

We present an approach for the detection of high polarization and hyperpartisan news articles. Our approach addresses stylistic and content related features. The latter are implemented by identifying n-grams that are typical for either a hyperpartisan or a more balanced perspective.

1.1 The Task

The goal of the SemEval2019 Hyperpartisan News Detection Task (Kiesel et al., 2019) is to build a system capable of classifying arbitrary articles either as non-hyperpartisan or hyperpartisan.

1.2 The Dataset

For building a classification system the organizers of the task provided several data sets extracted from different American news sites:

a) a set of 600.000 articles for training (classification: by publisher's general orientation)

b) a set of 150.000 articles for validation (classification: by publisher's general orientation)

c) 645 training articles (classified individually by humans using a crowd sourcing approach)

d) validation set (classified by publisher, unknown size as this data set has been hidden during the task)

e) validation set (classified individually, unknown size as this data set has been hidden during the task)

All data - the articles themselves as well as the ground truth data - is provided in an proprietary but simple and well parsable XML format defined by the task owners. The individual data records includes a globally unique ID, the title, the source URL and publication time. Additionally the ground truth data for a) and b) contains information about a left-right bias of the publisher in general.

2 Related Work

Not so much research has been done in regard to hyperpartisan news detection. Other work is primarily addressing related fields such as ideology detection, fake news detection. For example (Hutto et al., 2015; Hamborg et al., 2018) addresses identification and quantification of media bias. (Iyyer et al., 2014) addresses political ideology detection using recursive neural networks. (Rashkin et al., 2017) is analyzing language in fake news for automated political fact-checking. One interesting work directly targeting hyperpartisan news is the work of (Potthast et al., 2018) identifing hyperpartisan news articles via style.

3 Methodology

3.1 PoS Tagging

As we hypothesize that not all parts-of-speech are equally important for distinguishing hyperpartisan and non-hyperpartisan articles, we lemmatized and pos-tagged the dataset.

Initially, we experimented with the TreeTagger (Schmid, 1994), however since this turned out not to be sufficiently robust for the noisy input data, which included encoding errors as well as portions of JavaScript code, we later adopted the Stanford CoreNLP tagger (Manning et al., 2014).

3.2 Feature Extraction

We used a total of 108 features for our experiments. The next sections discusses our features in detail.

3.2.1 Linguistic Complexity and Style Features

Basic complexity: We hypothesize that hyperpartisan texts are stylistically less complex than non-hyperpartisan texts (Potthast et al., 2018), hence we implemented a number of features measuring linguistic complexity. We measure the distribution characteristics of paragraph lengths, sentence lengths and word lengths. Individual features were derived from that data like minimum, maximum, variance, mean, et cetera.

Number of words in main part-of-speech categories: We collect the number of verbs, nouns, adjectives and adverbs in articles and derive distribution features from it. While not every part-of-speech category will have the same importance we rationalize that at least the distribution characteristics of nouns, adjectives and adverbs could be a style hint for hyperpartisan or non-hyperpartisan.

Simple form of lexical density: As "lexical density" we here consider the ratio of words being not part of the NLTK stop-words in comparison to the total number of words. The idea behind this feature is to detect articles with lower or higher information character and take some stylistic aspect into account.

Huffman compression ratio: Our huffman compression feature is used with similar intention. First: The general idea behind huffman compression (Huffman, 1952) is to build a dictionary of words ordered by frequency in a binary tree. This is done in such a way that in the end high frequency words can be encoded with a shorter bit code than low frequency words. The rationale in our approach here is to perform a compression of individual articles: The better this compression works the more an author of an article reuses his own words. The more difficult this compression is, the higher is the variety of words used by an author. For speed reasons we intentionally do not perform a full compression here but build a huffman compression tree and then estimate the size the indices would take in a full compression. We then put this information into relation to the total number of tokens of an article and use this as a feature.

Readability scores: A set of readability scores is used. Readability scores express the simplicity of a text in various different ways - at least to some extent - as well as give a rough judgement for the reading competence level of an audience required to understand the text. We implemented features based on four readability scores: ARI (Smith and Senter, 1967), Coleman-Liau (Coleman and Liau, 1975), Flesch-Kincaid (Kincaid et al., 1975) and Gunning-Fog (Gunning, 1952).

Vocabulary variety: The vocabulary variety classifier is calculating the ratio of uniquely used words in relation to the total number of words. This way this classifier assists in judging the complexity of the text in an article as well.

3.2.2 Arousal vs. Rationality

Distribution parameters of business words: We assume that news articles addressing business related topics are inherently not particularly hyperpartisan. Based on manual inspection of training data, we therefore created a list of 27 words, which we consider to be expressing business related topics, for example "sales", "growth", "CEO", "opportunity", "revenue", "Q1", "shareholder" and similar terms.

Distribution parameters of words of disgust: In a similar way to business words the corpus lemmas are judged whether they express some kind of disgust. For this purpose a hand picked vocabulary of 246 words had been created from publically available online dictionaries such as LEO (LEO), Wictionary (Wictionary) and similar that genuinely express some kind of disgust. Though this dictionary likely is not complete the assumption is, that it gives a general insight into whether a writer expresses disgust at least to some extend, e.g. "discord", "rupture", "distaste", "scandalous" or even words like "rotten". We intended here not to detect only archetypical words such as “awful” but also more uncommon words that might typically not be seen in news so frequently. The rationale behind this is that we noticed the phenomenon of hyperpartisan authors to attempt to use a more vivid and strong language with sometimes less common words.

Cardinal number ratio: Detecting cardinal numbers is another feature addressing very specific aspects of articles: The idea behind this feature is that more fact-based communication might more likely make use of numbers in order to express and proof their positions. While we can not check the truth of claims involving cardinal numbers we at least try to detect the quantity of such claims.

Pronouns before “need” and “must”: Two feature detectors address pronouns directly proceeding the words “need” or “must”. We noticed hyperpartisan articles where the authors directly address the reader and give advice how society should proceed. This is done in an inclusive way, so sequences like “we must (do sth)” or “we need (to do sth)” could be observed.

3.2.3 Content Features

To address content specifically we implemented features derived from the provided test data itself, though this way these features can cover only limited and existing content.

Attributively used adjectives: According to the theories behind framing in psychology, political influence can be produced by repeating specific kind of wordings (Wehling, 2016). We noticed that this technique seems to be used sometimes quite extensively by authors of more extreme positions in recent years as they have a quite unchanging perspective about topics, persons and events. For example in the manually classified data the term “jewish” is used to characterize a following noun about five times more often in hyperpartisan than non-hyperpartisan articles, “holistic” about 30 times and “immediate” only about a third of the times compared to non-hyperpartisan news. Based on this phenomenon a dictionary of adjectives which discriminate a following noun have been extracted from non-hyperpartisan and hyperpartisan pos-tagged training data in a separate processing process, resulting in 2720 adjectives for our use. Our feature is then measuring whether more non-hyperpartisan or hyperpartisan use of such adjectives can be observed in an article.

Lemma-bigram similarity scores: While our attributively-used-adjectives-feature focuses on the adjectives themselves and is therefore a single word feature, we additionally used lemma based bigram features. We extracted all bigrams in sentences for a window of four tokens from the manually tagged training data (and for experiments from the larger data set) and associated them with either non-hyperpartisan or hyperpartisan labels. For example the lemmas “obama” directly followed by “administration” appear significantly more often in hyperpartisan than non-hyperpartisan articles. It’s even more extreme with “obamacare” and “act”, sequences of “bad” and “happen” or “disastrous” and “war”: The latter having even no mentions at all in non-hyperpartisan articles. Interestingly some bigrams are less characteristic as one would expect: For example “illegal” and “immigration” is used quite frequently by both classes. Again other bigrams seem to be more typical for non-hyperpartisan news articles, e.g. “fake” and “story”.

For our implementation we determined the relation of how often either hyperpartisan and non-hyperpartisan bigrams appeared per paragraph:

$$f = (nH - nNH)/(nH + nNH) \qquad (1)$$

where *nH* and *nNH* refer to the number of hyperpartisan/non-hyperpartisan bigrams. This value will be positive or negative depending on the surplus of non-hyperpartisan vs. hyperpartisan bigrams encountered in unseen text. We do this for directly adjacent lemmas, for two lemmas skipping one token, two tokens and three tokens and calculate the four medians so that we end up with a set of feature values, each one expressing content similarity to our reference data.

3.3 Machine Learning

We built two different models by training a support vector machine with an rbf-kernel (libsvm). The first one is based only on the stylistic features and has been submitted for the first evaluation run of the task, the second one is based only on the content features which has been submitted for the second evaluation run of the task.

For training of the first model 100.000 articles have been selected by random with stratified sampling, arriving at 25.000 articles classified as hyperpartisan left, 25.000 hyperpartisan right and 50.000 non-hyperpartisan. Selecting a sub-

	Dataset	Acc	Prec	Rec	F1
M1	by-pub.	0.505	0.503	0.949	0.657
M2	by-pub.	0.537	0.530	0.658	0.587
M2	by-art.	0.671	0.654	0.729	0.689

Table 1: Results, Model 1 and Model 2 with validation dataset used, accuracy, precision, recall and F1 score.

set of the available articles was necessary as part-of-speech tagging with first the TreeTagger and then the CoreNLP tagger took quite some time to complete. As mentioned before we ran into some tagging problems because of errors in the corpus data and limited capabilities of the existing Python adapters for CoreNLP. Additionally we encountered some problems with larger amounts of data which surprisingly caused crashes in the C implementation of the SVM (NuSVC of sklearn) for unknown reasons. So in the end we limited ourselves to these 100.000 random articles to cope with these difficulties.

As we recognized during our work that the training data classified by-pubisher was - by nature - not so accurately labeled, we train our second model on the gold standard data with 645 manually labeled articles to avoid any noise for our features as much as possible. For this model we used only the content features.

4 Results and Conclusions

To train our models we used the provided training data "by-publisher" and "by-article" as described in the last section. Evaluation runs have then been performed on the validation data "by-publisher" and "by-article" (which were hidden during the duration of the shared task). The results can be seen in table 1.

Model 1 (which was trained on the 100.000 randomly picked articles focusing on style features) was tested against the validation data labeled by-publisher. Model 2 (which was trained on the 645 articles focusing on content features) was tested against the validation data labeled by-publisher and the validation data by-article in two separate runs. Our model 1 achieved better results than model 2 during our evaluation runs on the by-publisher data. It has been selected by the organizers for ranking in the leader board.

Validation showed that our first model exhibits a trend to judge articles too easily as being hyperpartisan. Though our second model exhibits a trend to more easily classify articles as hyperpartisan as well, this effect is not that strong.

Our second, content feature model did not perform that well on the test data labeled by-publisher than the first, the style-based model. Interestingly it performed better on evaluation data labeled by-article. As our training data of 645 articles for that model is small the second model likely suffers from overfitting.

5 Further Work

In this paper we have presented a binary classification system that assign labels "non-hyperpartisan" and "hyperpartisan" for articles. While we could achieve some results in that field we still think that more work is needed here.

Results of competing teams in the SemEval2019 shared task indicate that our current approach has not yet been explored to full extent in that regard: Better classification could be possible. We assume that additional effort should be taken in selecting more and better style and content features. Though stylistic approaches seem to be promising – comp. (Potthast et al., 2018) – we assume that future work should focus more on content and empathic perception of the content by the reader. For example sentiment could be taken into consideration as news articles tend to have different point of views on different topics. As there exists a variety of different sentiment tools of varying quality experiments need to be performed to explore possibilities of improving our models. Attempts in this regard have been undertaken by ourselves already but could not be completed for this shared task. Additionally it would be interesting to combine both approaches, something we were not able to explore sufficiently during this shared task.

Acknowledgments

This work was funded by the ministry of science and culture of Lower Saxony ("Holen und Halten").

References

M. Coleman and T. L. Liau. 1975. A computer readability formula designed for machine scoring. In *Journal of Applied Psychology*, volume 60(2), pages 283–28.

Carroll Doherty. 2017. Key takeaways on americans growing partisan divide over political values. http://www.pewresearch.org/fact-tank/2017/10/05/takeaways-on-americans-growing-partisan-divide-over-political-values/. Accessed: 2019-02-21.

Robert Gunning. 1952. *The technique of clear writing*. New York: McGraw-Hill.

Felix Hamborg, Karsten Donnay, and Bela Gipp. 2018. Automated identification of media bias in news articles: an interdisciplinary literature review. *International Journal on Digital Libraries*.

D. A. Huffman. 1952. A method for the construction of minimum-redundancy codes. *Proceedings of the IEEE, formerly Proceedings of the IRE*, 40(9):1098–1101.

C.J. Hutto, Dennis Folds, and Scott Appling. 2015. Computationally detecting and quantifying the degree of bias in sentence-level text of news stories. In *The First International Conference on Human and Social Analytics*, pages 30–34.

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Ps Resnik. 2014. Political ideology detection using recursive neural networks. In *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, volume 1, pages 1113–1122.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. Semeval-2019 task 4: Hyperpartisan news detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

J. P. Kincaid, R. P. Fishburne, R. L. Rogers, and B. S. Chissom. 1975. Derivation of new readability formulas for navy enlisted personnel. Technical report.

LEO. Leo online dictionary. https://dict.leo.org. Accessed: 2019-02-21.

libsvm. SVM implementation library libsvm. https://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html. Part of Scikit-Learn; Authors: Chih-Chung Chang, Chih-Jen Lin; Accessed: 2019-02-21.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A stylometric inquiry into hyperpartisan and fake news. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 231–240. Association for Computational Linguistics.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2931–2937. Association for Computational Linguistics.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/. Accessed: 2019-02-21.

E. A. Smith and R. J. Senter. 1967. Automated readability index. Technical report.

E. Wehling. 2016. *Politisches Framing: Wie eine Nation sich ihr Denken einredet - und daraus Politik macht.* Ullstein.

Wictionary. Wictionary, the free dictionary. http://en.wictionary.org. Accessed: 2019-02-21.

Rouletabille at SemEval-2019 Task 4: Neural Network Baseline for Identification of Hyperpartisan Publishers

Jose G. Moreno and **Yoann Pitarch** and **Karen Pinel-Sauvagnat** and **Gilles Hubert**
IRIT / University of Toulouse
France
{jose.moreno, yoann.pitarch, karen.sauvagnat, gilles.hubert}@irit.fr

Abstract

This paper describes the Rouletabille participation to the Hyperpartisan News Detection task. We propose the use of different text classification methods for this task. Preliminary experiments using a similar collection used in Potthast et al. (2018) show that neural-based classification methods reach state-of-the art results. Our final submission is composed of a unique run that ranks among all runs at 3/49 position for the by-publisher test dataset and 43/96 for the by-article test dataset in terms of Accuracy.

1 Introduction

Printed press have been in the last decades the main way to access to news in written format. This tendency is changing with the appearance of online channels but usually the main factors of the journalistic content generation are still there: events, journalists, and editors. One of the problems of the generation of this content is the influence of each factor in the veracity of the generated content. Two main factors may influence the final view of an article: writer's preferences and affiliation of the editor house.

Identifying partisan preferences in news, based only on text content, has been shown to be a challenging task (Potthast et al., 2018). This problem requires to identify if a news article was written in such a way that it includes an overrated appreciation of one of the participants in the news (a political party, a person, a company, etc.). Despite the fact that sharply polarized documents are not necessarily fake, it is an early problem to solve for the identification of fake content. A recent paper (Potthast et al., 2018) claims that stylometric features are a key factor to tackle this task.

In this paper, we present the description of our participation to the Hyperpartisan classification task at SemEval-2019 (Kiesel et al., 2019). This task was composed of two subtasks, the first consist to identify hyperpartisan bias in documents classified by its individual content (bias of the writer or by-article category) and the second by the editorial house that published the article (bias of the editorial house or by-publisher category) as depicted in Figure 1[1]. To address this problem, we experimented with well-known models based on deep learning (Honnibal and Montani, 2017; Kim, 2014). They achieve state-of-the-art results on a publicly available collection (Potthast et al., 2018), showing that neural models can effectively address the task of hyperpartisan detection without including stylometric features. Our final submission ranked in the top-3 for the by-publisher category, and 43/96 for the by-article category (or 21/42 in the official ranking).

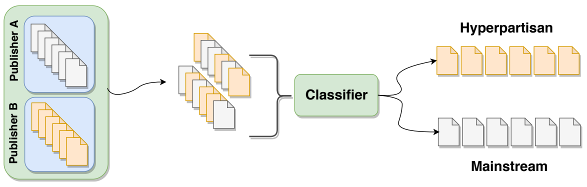


Figure 1: Publisher-based pipeline performed in training phase. During testing, different publishers were used and labels were unknown.

2 Classification Models

We have considered that the hyperpartisan classification task can be addressed as a binary classification task where only two classes ('hyperpartisan' and 'mainstream').

Three different models were considered for our participation. The first of them is based on a classical document-level representation and the

[1]More details of the dataset construction can be found in Kiesel et al. (2019)

other two are based on word-level representations through the use of word embedding. All of them can be seen as baselines and no specific adaptation to the dataset[2] was performed.[3]

2.1 TF-IDF + Adaboost

For this model we represented our documents using the classical TF-IDF representation. Finally, the Adaboost classifier (Freund and Schapire, 1997) is used under the default configuration. Note that this is a very basic baseline, as it does not use recent representation techniques such as word embeddings.

2.2 SpaCy Model

In this case, we used the SpaCy (Honnibal and Montani, 2017) library[4]. We used the text categorisation algorithm implemented in SpaCy which is based on the hierarchical attention network proposed in Yang et al. (2016). The main improvement to the original model is the use of hash-based embeddings. We only defined two hyper-parameters for the model: number of epochs and dropout rate. These parameters were set to 3 and 0.2, respectevelly.[5]

2.3 Convolutional Model

We also tested the neural classification model proposed by Kim (2014). This model uses convolutional neural networks that are finally reduced to a binary classification. This method is known as a highly competitive classification model for short documents. As SpaCy, this model is based on word embeddings representation. However, in this case we preferred to use the pre-calculated embeddings of GloVe (Pennington et al., 2014). Hyper-parameters were defined using the training data.

3 Experiments and Results

3.1 Experimental Setup

Experiments were performed using two collections, the ACL2018 collection (Potthast et al., 2018) and the SemEval19 collection (Potthast et al., 2019). The first collection is composed of 1627 articles including 801 hyperpartisan and 826 mainstream manually annotated documents. As this collection is not originally split in training-test sets, results are presented using cross-validation. The second collection was split in train, validation, and test sets for the by-publisher category, and in train and test for the by-article category as presented in Table 1. Results in this second collection are exclusively calculated using the TIRA evaluation system (Potthast et al., 2019).

	training	validation	test
by-article	645	-	628
by-publisher	600000	150000	4000

Table 1: Number of documents used for training, validation, and test used in the SemEval19 collection.

In order to determine the best configuration to our participation using the SemEval collection, we decided to perform experiments and fix hyperparameters using the ACL2018 collection.

3.2 Results in the ACL2018 Collection

Table 2 reported results of the 3 classification models presented in section 2 (lines labelled TF-IDF+Adaboost, SpaCy and CNN-Kim), as well as results of the approach presented in Potthast et al. (2018) (line labelled ACL18), on the ACL2018 collection.

We only used the first fold produced by the authors' code[6]. As our results are not directly comparable with the values reported in Potthast et al. (2018), we re-evaluated their approach on this single fold.

Values of the three F-measures were calculated with sklearn[7]. Note that in binary classification, micro F-measure values are equivalent to accuracy values.

Two state-of-the-art models (SpaCy and Kim (2014)) outperform the approach presented in Potthast et al. (2018), showing that stylometric features are probably not necessary for the task.

3.3 Results in the Semeval2019 Collection

Experiments on the official collection were performed through the use of TIRA (Potthast et al., 2019)[8]. As our previous experiments have not shown clear improvement with the convolutional model, we submitted our official runs using

[2]Different to the classical training of the involved classifiers.

[3]Further experiments were performed using network-based models but as results did not show improvement in an existing collection, we decided to not include these results.

[4]https://spacy.io/

[5]We based on SpaCy's guidelines.

[6]https://github.com/webis-de/ACL-18

[7]https://scikit-learn.org/

[8]https://www.tira.io/task/hyperpartisan-news-detection/

	F-measure		
	macro	accuracy /micro	weighted
ACL18	0.7605	0.7509	0.7480
TF-IDF + Adaboost	0.7069	0.7130	0.7039
SpaCy (dp =0.2, epochs=3)	0.8087	0.8091	0.8081
CNN-Kim	**0.8273**	**0.8306**	**0.8290**

Table 2: Macro, micro and weighted F-measure for the ACL2018 collection.

	accuracy/micro	f1
top1	**0.7060**	0.6825
top2	0.6998	0.6587
our (rank 3/49)	0.6805	**0.7213**
top4	0.6640	0.7061

Table 3: Official results for the by-publisher test dataset.

SpaCy: it can be seen as an 'easy-to-implement' but strong baseline. The same model was trained on the by-publisher training set for both submissions (on the by-publisher and by-article dataset).

Tables 3 and 4 respectively present official results on the by-publisher[9] and the by-article datasets.

One can see that relative results (i.e. regarding the official ranking) are strongly better on the by-publisher dataset than on the by-article one. This can be easily explained by the fact that collections were differently annotated.

If we now compare accuracy scores of the SpaCy model between the ACL2018 collection and the SemEval2019 one, we can notice a decrease in performance (0.6640 vs 0.8091 on the by-publisher dataset for example), leading us to think that there exist some differences between the two collections. Both collections seem to be complementary for the evaluation of hyperpartisan detection.

Another important observation is that the SpaCy model performs remarkably well on the by-publisher set, although not specifically tuned for the hyperpartisan detection task. Indeed, we are ranked first on the F1 metric, and 3rd on the Accuracy one. Some other experiments are needed to get a fine-tuned model for the task, but this version can already be considered as a strong baseline for the by-publiser subtask.

[9]https://www.tira.io/task/hyperpartisan-news-detection/dataset/pan19-hyperpartisan-news-detection-by-publisher-test-dataset-2018-12-12/ last visit 19/02/2019.

	accuracy /micro	f1
top1	**0.8217**	0.8089
top2	0.8201	**0.8215**
top3	0.8089	0.8046
our (rank 43/96)	0.7245	0.6905

Table 4: Official results for the by-article test dataset.

4 Conclusion

Our experiments and participation to the Hyperpartisan task led us to conclude that:

- stylometric features seem not to be necessary to achieve state-of-the-art results for hyperpartisan detection in the ACL2018 collection. This deserves a set of extra experiments to better understand the real contribution of stylometric features when combined with strong representations/classifiers to validate the work of Potthast et al. (2018).
- a state-of-the-art classification model in its default configuration (SpaCy) can be considered as a strong baseline for next experiments. Indeed, SpaCy is top-ranked according to the F1 metric on the by-publisher dataset. One question is thus now if other top-ranked approaches are also from the text classification literature or dedicated ones.

References

Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. https://spaCy.io.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. TIRA Integrated Research Architecture. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 231–240. Association for Computational Linguistics.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Spider-Jerusalem at SemEval-2019 Task 4: Hyperpartisan News Detection

Amal Alabdulkarim
Department of Computer Science
Columbia University
amal.a@columbia.edu

Tariq Alhindi
Department of Computer Science
Columbia University
tariq@cs.columbia.edu

Abstract

This paper describes our system for detecting hyperpartisan news articles, which was submitted for the shared task in SemEval 2019 on Hyperpartisan News Detection. We developed a Support Vector Machine (SVM) model that uses TF-IDF of tokens, Language Inquiry and Word Count (LIWC) features, and structural features such as number of paragraphs and hyperlink count in an article. The model was trained on 645 articles from two classes: mainstream and hyperpartisan. Our system was ranked seventeenth out of forty two participating teams in the binary classification task with an accuracy score of 0.742 on the blind test set (the accuracy of the top ranked system was 0.822). We provide a detailed description of our preprocessing steps, discussion of our experiments using different combinations of features, and analysis of our results and prediction errors.

1 Introduction

Fake news on various online media outlets misinform the public and threaten the integrity of journalism. This has serious effects on shaping public opinions on controversial topics such as climate change, and swaying voters in political elections. Yellow press existed long before the digital age but had limited reach when compared to mainstream press. However, with the introduction of social media, news that are extremely biased (hyperpartisan) tend to spread more quickly than the ones that are not (Vosoughi et al., 2018). Therefore, there is a need for automatic detection methods of hyperpartisan news. Computational methods for fighting fake news mainly focus on automatic fact-checking rather than looking at writing styles of news articles (Potthast et al., 2018). SemEval-2019 Hyperpartisan News Detection shared task aims to study the ability of a system to detect if a given article exhibits a hyperpartisan argumentation writing style to convince readers of a certain position. The shared task introduces a binary classification task of classifying an article into one of two possibilities: mainstream or hyperpartisan. The data for the shared task was introduced by (Kiesel et al., 2019) and consists of 645 of articles from mainstream, left-wing, and right-wing publishers. The articles from both left-wing and right-wing publishers were labeled as hyperpartisan.

The baseline system to detect hyperpartisan developed by (Potthast et al., 2018) uses Unmasking (Koppel et al., 2007) and was trained on 1,627 of articles. The articles are from nine publishers in the US: three mainstream (ABC News, CNN, and Politico), three left-wing (Addicting Info, Occupy Democrats, and The Other 98%), and three right-wing (Eagle Rising, Freedom Daily, and Right Wing News). Their model had a best accuracy of 75% by using stylistic features. However, their model is not directly comparable with ours since the dataset for the shared task is different.

In the following sections, we describe our system for identifying hyperpartisan news articles as part of our participation in the Hyperpartisan New Detection shared task in SemEval 2019 (Kiesel et al., 2019).

2 System Description

We trained a support vector machine model on a feature vector representing each article in our training dataset. To develop this model, we processed the dataset and analyzed different features and feature combinations.

2.1 Pre-processing

The training dataset contained 645 articles that include 238 (37%) hyperpartisan and 407 (63%) mainstream (Kiesel et al., 2018). The test dataset is 628 articles (314 from each class).

We clean the articles and titles from punctuation marks, stop words, none alphabetical characters, lemmatized and tokenized using the Natural Language Toolkit (NLTK) (Bird et al., 2009) . After that, those tokens are processed using the TF-IDF vectorizer in sci-kit-learn (Pedregosa et al., 2011) and stored as a vector.

2.2 Feature Extraction

The features we chose to extract from the articles, include the following:

1. *Words vector.* After pre-processing all the unigrams in the articles and the titles are stored in a TF-IDF vector.

2. *Linguistic Inquiry and Word Count (LIWC) features.* The words in every article and titles that are part of any dimension of the Linguistic Inquiry and Word Count (LIWC) dictionary. LIWC analyzes text by using a dictionary of the most common words and word stems. The dictionary is organized into different categories, some of which are affect words and function words. (Pennebaker et al., 2012) are counted and stored in a sixty-three dimensional TF-IDF vector.

3. *Punctuation.* The punctuation marks in the title and articles were grouped into six different categories and then counted and stored separately from the article and then stored in a six-dimensional TF-IDF vector. Because we were specifically interested in exclamation marks, question marks and quotations we let those three punctuation marks have their independent counts in the vector. The other three dimensions are colons and comma, dot, and parenthesis.

4. Article structure features:*Paragraphs, quotes, and external links* are counted and stored in a 3-dimensional vector.

5. *Emotion features.* The emotional content in the articles is captured using the NRC emotions lexicon (Mohammad and Turney, 2013). After counting the words in each emotion category, we store the counts in a ten-dimensional vector, where the elements represent anger, anticipation, disgust, fear, joy, negative, positive, sadness, surprise and trust.

After pre-processing and extraction, we experiment with different groupings of these features in our model to see which group of features is most effective for the given task. The next section discusses those feature combinations.

Features	Title				Article					F1
	W	L	P	E	W	L	P	E	S	
1	x	x	x	x						0.48
2					x	x	x	x		0.75
3					x	x	x	x	x	0.74
4	x	x	x	x	x	x	x	x	x	0.72
5	x				x					0.76
6		x	x	x		x	x	x	x	0.71
7					x	x	x		x	0.76
8	x	x		x	x	x		x		0.74
9		x				x				0.48
10			x				x			0.48
11				x				x		0.70
12	x				x	x	x		x	0.76

Table 1: Different feature combinations (W: Words vector, L: LIWC features, P: Punctuation Marks, E: Emotions and S: Articles Structure) and their weighted F1 score on the local validation set.

2.3 Feature Selection

Now that we have the extracted features we began grouping them and testing them in our model. We tested the different combinations of these features as shown in Table 1. In these experiments, the most effective combination of features was number 7 (the word tokens, LIWC, punctuation marks, and the article structure), number 5 (the word tokens of the article and title) and number 12 (all the features in 5 and 7). The other title features did not provide a good contribution to the model as we expected.

2.4 Model

The model[1] is constructed using a sci-kit-learn pipeline with two main steps. The first part is a dimensionality reduction using latent semantic indexing (Manning et al., 2008) using Truncated Singular value decomposition (SVD). The primary goal of using an SVD is to lower the rank of the feature matrix by merging the dimensions associated with terms that have a similar meaning. The second step is a linear support vector machine (SVM) model used in default settings, which takes as input the output of the

[1]https://github.com/amal994/hyperpartisan-detection-task

SVD. The SVM is useful in high dimensional spaces and when the number of features is higher than the number of articles in the dataset.

3 Results

In this section, we will review the results of our model and show its performance on a local validation set of size 129 articles (48 hyperpartisan, 81 not hyperpartisan) and the test set on TIRA (Potthast et al., 2019).

Measure	Validation Set	Test Set
accuracy	0.767	0.742
f1-score	0.767	0.709
precision	0.767	0.814
recall	0.767	0.627
true positives	26	197
true negatives	73	269
false positives	8	45
false negatives	22	117

Table 2: Main task classification results of the local validation and test datasets.

3.1 Main Task

For the main task, identifying hyperpartisan articles from a dataset of manually labeled articles, we created a local validation set, by partitioning the by-article dataset into a training and validation sets while keeping the split ratio equal in both. We do not report any results on the by-publisher datatset as we found class mismatch for some articles across the two datasets (i.e. some articles are labeled mainstream in the by-article and hyperpartisan in the by-publisher). Therefore, we decided to focus on the more accurately labeled dataset (the by-article), which is also the one used for share-task leaderboard ranking.

Model	Accuracy	
	validation set	test set
SVM	0.767	0.742
Ensemble	0.829	0.640
Ensemble-RNN	0.76*	0.694

Table 3: Classification results of various models. Ensemble-RNN model was tested using cross-validation so it is not directly comparable with the other two models in the validation scores.

In Table 2, we show the classification report of the SVM model after running on a local validation set and the official test set. When we tested the SVM model locally, it gave a high f1-score 0.767 which is the measure we relied on locally because the data was not balanced.
On the task leaderboard, tested on the test set in TIRA, the model ranked 17 among the 42 participating teams, with an accuracy of 0.742.

We also experimented with other machine learning models and compared them with our SVM model. We developed an ensemble model that consists of an SVM classifier, a Gradient Boosting Classifier and a Bagging Classifier with a decision tree as its base estimator. But that classifier only scored 0.64 accuracy on the test set, even though it scored 0.829 accuracy on the validation set. We also added an RNN classifier that uses ELMO embeddings (Peters et al., 2018) to the previously described ensemble model. That model increased the accuracy on the test set by a small value 0.694 but did not outperformed the SVM model.

3.2 Meta Learning

We also participated in the meta-learning sub-task, the task is to use all of the predictions from all of the participating teams classifiers as an input and come up with a meta classifier.

The dataset we were given is a list of predictions from all the participating classifiers and the gold labels for each article in this list.
The model we developed builds on the idea of a weighted majority algorithm but with changes to how the weights are being calculated. So instead of dividing by the total number of elements to calculate the weights, we have two separate weights, one for each class, and then we calculated those two weights for each classifier using equation 1 where H in the equation corresponds to the class (0 or 1), c is the classifier and y is the true label.

$$w(c, H) = \frac{\sum_i^n \mathbb{1}(y = H \wedge c(x) = H)}{\sum_i^n \mathbb{1}(y == H)} \quad (1)$$

This classifier had a validation accuracy of 0.899 and the baseline majority vote classifier 0.884. The model has only a slight advantage in its accuracy which is beneficial for the competition. Even though when used in real life the difference between the two accuracies is negligible.

4 Discussion

We can observe from the results in Table 1 that the TF-IDF features of articles and titles are the most useful for this task. They consistently have the highest accuracy score when combined with other features or when used alone as shown in feature set 5 in Table 1. This shows that it was hard for LIWC features by themselves to capture any linguistic patterns that correlate with hyperpartisan news. The superiority of TF-IDF could be due to trends related to a certain domain or publisher rather than to a general hyperpartisan trend. In order to examine the ability of other approaches to detect hyperpartisan news articles, we developed two other models. An Ensemble model of three models and an Ensemble-RNN model both described in Section 3.1. Both models scored almost as good as the SVM on the validation set (Ensemble-RNN model) or better than the SVM (Ensemble model). However, both scored significantly lower than the SVM model on the blind test set. The Ensemble-RNN model included a neural network which was trained on our small training set of 645 articles. Given the huge drop between validation and test scores, especially for the ensemble mode which dropped from 0.83 to 0.64, this indicates an overfitting on the training data. Although the deep learning models were not trained for more than five epochs to avoid overfitting, they were not able to learn beyond what was seen in the training data and were possibly memorizing the data. The complexity and subjectivity of the annotation task could have made it harder for the model to classify articles. We were also dealing with imbalance class sizes which made the model learn to predict one class better than the other. As for the meta-learning experiment, we followed a class-based weighted majority approach, where the classifiers that are better in classification of one class were given a higher weight for that class predictions and lower weight for their predictions in the other class. However, this approach only had a one-point improvement over the baseline. We analyzed the prediction errors of the SVM model to further understand what causes the model to make a wrong prediction.

4.1 Error Analysis

We looked more closely at four examples: one correct and one wrong prediction from each class. The first example is an article from Fox News about the 2016 US presidential elections [2]. This article was labeled as mainstream and was predicted correctly by our model. Although the article was predicted correctly by the classifier, it was not clear to us why this article was labeled as mainstream as it has a somewhat one-sided view to its story and thus could be labeled as hyperpartisan. This points out the uncertainty or noise in the annotation of some data points. The second mainstream article is from Yahoo! news and talks about Ivanka Trump [3]. It was wrongly classified as hyperpartisan by our model. This could be due to the fact that the content of the article is related to Trump, which appeared more in the hyperpartisan class in our dataset. The third article is labeled as hyperpartisan and predicted as such. It is an opinion piece from Online Athens about social justice [4]. It has phrases such as “so-called” and “Karl Marx would be so proud” which could’ve helped the model to use the learned TF-IDF features of the training data to make a correct prediction. The final article we looked at is from Real Clear Politics and talks about a joke made by Jimmy Kimmel [5]. This was wrongly classified as mainstream by the model which could be due to having structural features of mainstream articles (long article and no URLs). These four examples show that some of our lexical and structural features did not generalize well to the test set.

5 Conclusion

We presented an SVM model that detects hyperpartisan news articles with a 0.742 accuracy after it was trained on a total 645 articles from mainstream and hyperpartisan classes. This task was primarily challenging due to the complexity in labeling such articles, and differences in writing styles across domains, publishers and individuals. The small size of the training data along with the class imbalance also contributed to the complexity, which made it harder for the model to learn. We presented a summary of our experiments and analysis of our results and prediction errors.

[2]http://insider.foxnews.com/2016/10/14/
[3]http://www.yahoo.com/news/truck-ad-featuring-ivanka
[4]http://onlineathens.com/opinion/2017-10-19/
[5]https://www.realclearpolitics.com/articles/2017/09/22/

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Inc.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, David Corney, Payam Adineh, Benno Stein, and Martin Potthast. 2018. Data for PAN at SemEval 2019 Task 4: Hyperpartisan News Detection.

Moshe Koppel, Jonathan Schler, and Elisheva Bonchek-Dokow. 2007. Measuring Differentiability: Unmasking Pseudonymous Authors. *Journal of Machine Learning Research*, (8):1261–1276.

Christopher D. Manning, Prabhakar. Raghavan, and Hinrich. Schutze. 2008. *Introduction to information retrieval*. Cambridge University Press.

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a Word-Emotion Association Lexicon. *Computational Intelligence*, 29:436–465.

Fabian Pedregosa, Vincent Michel, Olivier Grisel OLIVIERGRISEL, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Jake Vanderplas, David Cournapeau, Gal Varoquaux, Alexandre Gramfort, Bertrand Thirion, Olivier Grisel, Vincent Dubourg, Alexandre Passos, Matthieu Brucher, Matthieu Perrot andÉdouardand, Anddouard Duchesnay, and FRdouard Duchesnay EDOUARDDUCHESNAY. 2011. Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot. Technical report, Parietal, INRIA Saclay.

James W Pennebaker, Roger J Booth, Ryan L Boyd, and Martha E Francis. 2012. Linguistic Inquiry and Word Count: LIWC2015 Operator's Manual. In *Applied Natural Language Processing*, pages 206–229. IGI Global.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. TIRA Integrated Research Architecture. In *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, page 231240.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science (New York, N.Y.)*, 359(6380):1146–1151.

Steve Martin at SemEval-2019 Task 4: Ensemble Learning Model for Detecting Hyperpartisan News

Youngjun Joo
Department of Computer Engineering
Yonsei University, Seoul, Korea
yj.joo@yonsei.ac.kr

Inchon Hwang
Department of Computer Engineering
Yonsei University, Seoul, Korea
ich0103@yonsei.ac.kr

Abstract

This paper describes our submission to task 4 in SemEval 2019, i.e., *hyperpartisan news detection*. Our model aims at detecting hyperpartisan news by incorporating the style-based features and the content-based features. We extract a broad number of feature sets and use as our learning algorithms the GBDT and the n-gram CNN model. Finally, we apply the weighted average for effective learning between the two models. Our model achieves an accuracy of 0.745 on the test set in subtask A.

1 Introduction

The proliferation of misleading information in the media has made it challenging to identify trustworthy news sources, thus increasing the need for fake news detection tools able to provide insight into the reliability of news contents. Since the spread of fake news is causing irreversible results, near-real-time fake news detection is crucial. However, knowledge-based and context-based approaches to fake news detection can only be applied after publication; they may not be fast enough (Potthast et al., 2017).

As a practical alternative, style-based approaches try to detect fake news by capturing the manipulators in the writing style of news content. This approach captures style signals that can indicate a decreased objectivity of news content and thus the potential to mislead consumers, such as hyperpartisan style. Hyperpartisan style represents extreme behavior in favor of a particular political party, which often correlates with a strong motivation to create fake news. Linguistic-based features can be applied to detect hyperpartisan articles (Potthast et al., 2017). Deep network models, such as convolution neural networks (CNN), applied to classify fake news detection (Wang, 2017). In this paper, we employ the stylometry-based approach and N-gram CNN model for detecting hyperpartisan news.

2 System Overview

For this task, we extract a broad number of features from the training data and then apply the classifier model to make predictions. Our system employs a gradient boosting decision tree (GBDT) model and N-gram CNN model. In subsequent sections, we describe data preprocessing, feature engineering and learning algorithms.

2.1 Data Preprocessing

Before applying the models, we need to do some transforming tasks of the article texts (i.e., *xml parsing, text tokenizing, stemming, lemmatization, and removing stopwords*) and extracting tasks of the internal and external links for each article. Apart from these tasks, we construct the bias domain dictionary from the *mediabiasfactcheck* site [1] to check the bias on the external linked domain in the article. For this ends, we crawled the top-level domain information from the sites corresponding to the five categories associated with hyperpartisan (e.g., *Left, Center, Least Biased, Right-center Bias, and Right Bias*) respectively.

2.2 Feature Engineering

Since hyperpartisan news is intentionally created for political gain rather than to report objective claims, they often contain opinionated and inflammatory language. Thus, it is reasonable to exploit linguistic features that capture different writing styles to detect hyperpartisan news. Linguistic features are extracted from the text content in terms of document organizations at a different level, such as characters, words, and sentences. Typical common linguisitic features are: *lexical*

[1] http://mediabiasfactcheck.com/

Type of Features	Feature Count
Count features	10
External link bias	3
Sentiment features	8
Readability features	14
Term features	44
Grammar transformation	45
Psycholinguistic features	54
POS tags	36
Word2vec features	301
TF-IDF	10,000

Table 1: Statistics of features.

features, including character-level and word-level features; *syntactic features*, including sentence-level features (i.e., *n-gram, POS tagging, etc.*).

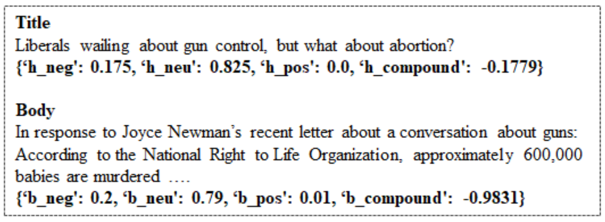
We start by extracting several sets of linguistic features. These feature sets are designed to capture hyperpartisan article from the training datasets. Overall we selected 515 binary features and TF-IDF features. Table 1 provides extracted features on the training dataset.

Basic count features: Previous works on fake news detection (Rubin et al., 2016) as well as on opinion spam (Ott et al., 2011) suggest that the use of punctuation is useful to differentiate deceptive from truthful texts. We construct a basic count feature set including various punctuation characters and other features.

External link bias: We extract bias counts based on the bias domain dictionary for each external linked domain in the article (i.e.,*hyperpartisan links count, non-hyperpartisan links count, and unknown links count*). To determine biases of the external links, we exploit a biased domain dictionary crawling from the *mediabasisfactcheck* site, which consists of five categories for top-level domains(i.e., *left, right, left-center, center, right-center*). The external link bias is counted as the hyperpartisan when the externally linked site is belonging to *left* and *right* among these categories.

Sentiment features: Our system used the VADER sentiment analysis tools [2] to generate sentiment features on the title and body of articles. The VADER not only tells about the *Positivity* and *Negativity* score but also tells us about how positive or negative a sentiment is as shown in Figure 1.

[2] https://github.com/cjhutto/vaderSentiment


Title
Liberals wailing about gun control, but what about abortion?
{'h_neg': 0.175, 'h_neu': 0.825, 'h_pos': 0.0, 'h_compound': -0.1779}

Body
In response to Joyce Newman's recent letter about a conversation about guns: According to the National Right to Life Organization, approximately 600,000 babies are murdered ....
{'b_neg': 0.2, 'b_neu': 0.79, 'b_pos': 0.01, 'b_compound': -0.9831}

Figure 1: An example of sentiment analysis.

Vocabulary richness and readability features: We also extract features indicating article understandability. These features include several vocabulary richness and readability scores, including the Brunet's Measure W, Hapax DisLegemena, Hapax Legomenon, Honores R Measure, Sichels Measure, Yules Characteristic K, Dale Chall Readability Formula, Flesch Reading Ease, Gunning Fog Index, Shannon Entropy, Simpson's Index etc[3]. Among this index, Simpson's index stems from the concept of biodiversity. We apply this index to measure the diversity of a text.

Simpson's Index $(D) = \sum(n/N)^2$
N = total number of words in a text
n = total number of unique tokens

Term features: Hyperpartisan news uses their language strategically despite the attempt to control what they are saying. This language occurs with certain verbal aspects and patterns of pronoun, conjunction, and negative emotional word usage. Based on this assumption, we extract term count features which count synonyms of several terms (e.g., to obtain the ORDER term Feature, we calculated the frequency of words such as *command, demand, instruction, prescription, order* in each article).

Grammar transformation: Analysis of the content-based approach is often not enough in predicting hyperpartisan news. Thus, we adopt language structure (syntax) to predict this task. We use spaCy tool [4] to transform news articles into a set of parse tree describing syntax structure.

Psycholinguistic features: For psycholinguistic features, we use the 2015 Linguistic Inquiry and Word Count (LIWC[5]) lexicon to extract the proportions of words that belong to the psycholinguistic categories. LIWC has two types of categories; the first kind captures the writing style

[3] https://en.wikipedia.org/wiki/Readability
[4] https://spacy.io
[5] http://liwc.wpengine.com/

of the author by considering features like the POS frequency or the length of the used words. The second category captures content information by counting the frequency of words related to some thematic categories such as affective processes(e.g., *positive emotion, negative emotion, anxiety, anger, sadness*), social processes (e.g., *family, friends, female references, male references*), etc. Regarding the use of this tool, we focus on the content information, and consequently, we decide to ignore the style categories.

Part-of-Speech (POS) tags: Syntactic features consist of function words and part-of-speech tags. Syntactic pattern varies significantly from one author to another. These features were extracted using more accurate and robust text analysis tools (i.e., part-of-speech taggers, and lemmatizers). In our system, we expand the possibilities of word-level analysis by extracting the utilities of features like POS frequency. For the extraction of syntactic features, we used NLTK POS tagger[1].

Word2Vec features : Recently, word representation model (e.g., *word2vec, GloVe*) based on neural networks which represents a word into a form of a real-valued vector have increased popularity (Mikolov et al., 2013). These approaches proved to be advantageous in many NLP tasks, such as Machine Translation, Question Answering, Document Classification, to name a few. We adopted a pre-trained 300-dimensional word vector [6] to create a vector representation of the article, with an average word2vec. Besides, we use the word2vec feature to extract the cosine similarity value between the news title and the text.

TF-IDF features: Finally, We extract unigrams, bigrams, and trigrams derived from the bag of words representation of each news article. To account for occasional differences in content length between train dataset and test dataset, these features are encoded as tf-idf values. We limit the number of features that the vectorizer will learn to 10,000 features.

2.3 Learning Algorithms

Based on the above multiple features, we explore several learning algorithms to build classification models. We adopt the average weighted value for effective learning between GBDT for the style-based and content-based features and the N-gram CNN model. (see Figure 2).

Layer	# of layers		hyperparameters
Embedding	1	l	5000
		d	300
Convoulution	3	m	[500,500,500]
		w	[3,4,5]
		w	max
Dense Layer	2	t	128
		o	2

l: max sequence length	d: embedding dimension
m: filter	w: kernel size
w: max-pooling	t: dense unit size
o: softmax	

Table 2: N-gram CNN model hyperparameters.

For deep learning model, we adopt N-gram CNN model proposed in (Shrestha et al., 2017). As shown in Figure 2 (right), the model receives a sequence of character n-gram as input. These N-gram are then processed by four layers: (1) an embedding layer, (2) a convolution layer, (3) a max-pooling layer, and (4) a softmax layer. We briefly sketch the processing procedure.

The network takes a sequence of character bigrams $x =< x_1, ..., x_l >$ as input, and outputs a multinomial over class labels as a prediction. The model first look up the embedding matrix to generate the embeddings sequence for x (i.e., the matrix C), and then pushes the embedding sequence through convolutional filters of three bigram-window sizes $w = 3, 4, 5$, each yielding m feature maps. We then apply the max-pooling to the feature maps of each filter, and concatenate the result vectors to obtain a single vector y, which then generate a prediction through the softmax layer.

Based on this model, we modified the network by adding a dense layer which helps detect hyperpartisan news features. After the experiment, the result shows that the character bigram CNN model outperforms the unigram CNN model. Table 2 summarizes the sizes of various parameters included in the N-gram CNN model. The official evaluation measure for subtasks A is *accuracy*.

Table 3

3 Experiments and Results

3.1 Datasets

The statistics of the datasets provided by SemEval 2019 task 4 (Kiesel et al., 2019) are shown Table

[1]https://www.nltk.org/
[6] https://code.google.com/archieve/p/word2vec/

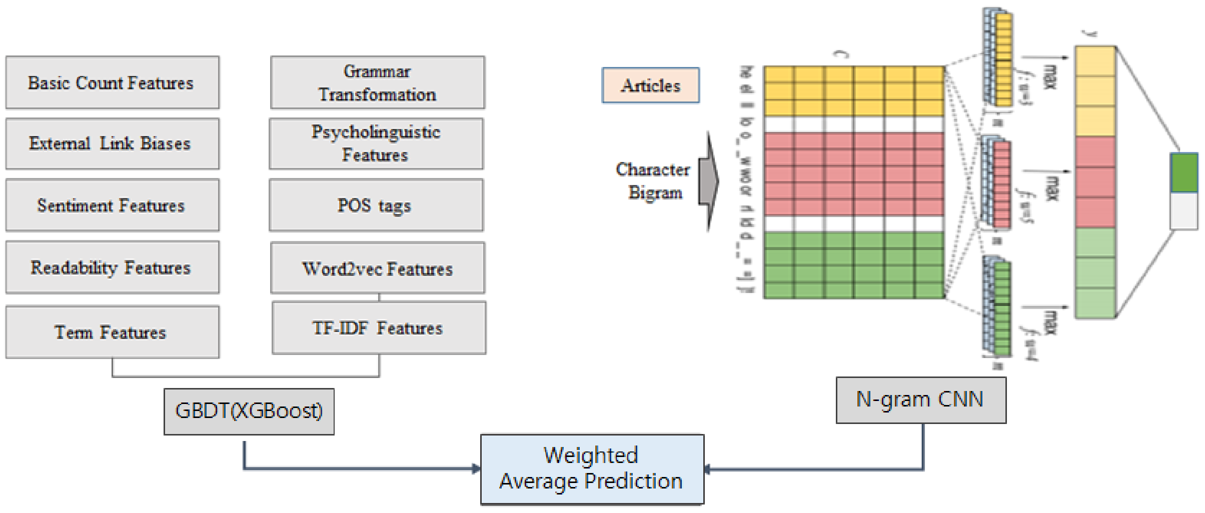


Figure 2: Hyperpartisan news detection model.

Subtask A	Hp(%)	NHp(%)
train (645)	238(36.9)	407(63.1)
Subtask B	Hp(%)	NHp(%)
train (600k)	300k(50%)	300k(50%)
valid (150k)	75k(50%)	75k(50%)

Table 3: Statistics of data sets in SemEval 2019 Task 4 Hp: hyperpartisan news; NHp: non-hyperpartisan news.

3.

3.2 Experiments on the Train Dataset

We conduct several experiments on each feature set to explore predictive separately. In these experiments, we use the GDBC (i.e., XGBoost) for the above feature set. For comparison with the N-gram model, we used the Char-level CNN model (Kim et al., 2016). The objective function was minimized through stochastic gradient descent over shuffled mini-batches with Adam(Kingma and Ba, 2014).

The performance is evaluated using 5-fold cross validation with accuracy and F-score. Table 4 lists the experimental results for each feature set on the training dataset. The prediction model through the incorporation of the entire feature showed higher accuracy than the prediction model for the individual feature.

3.3 Experiments on the Test Dataset

Our submission results to the subtask A on TIRA (Potthast et al., 2019)–the web service platform to facilitate software submissions into virtual machine– achieve an accuracy of **0.745** (precision: 0.853, recall: 0.592, F1: 0.6999). We ranked the 14th for subtask A in terms of accuracy. The prediction results of the test data are lower than the results of the training set, especially gains huge gap between precision and recall score.

Features (# of features)	Acc	F1
Count features (10)	0.6977	0.60
External link bias (3)	0.6512	0.60
Sentiment features (8)	0.6124	0.61
Readability features (14)	0.7442	0.74
Term features (44)	0.6512	0.65
Grammar transformation (45)	0.7829	0.78
Psycholinguistic features (54)	0.7984	0.79
POS tags (36)	0.7132	0.72
Word2vec features (301)	0.7752	0.77
TF-IDF (10,000)	0.7364	0.73
Char CNN (unigram)	0.7442	0.73
N-gram CNN (bigram)	0.7752	0.78
All Features (train dataset)	**0.8450**	**0.84**
All Features (test dataset)	**0.7450**	**0.70**

Table 4: Experimental results on the subtask A dataset.

4 Conclusion

Using a combination of the style-based approaches, the content-based approaches, and the N-gram CNN model, we construct the model for detecting hyperpartisan news. For this ends, we extract a broad number of linguistic features and employ GBDT model to make predictions. Finally, we adopted the weighted average value for effective learning between the two models.

References

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. Semeval-2019 task 4: Hyperpartisan news detection. In *In Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 309–319. Association for Computational Linguistics.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. TIRA Integrated Research Architecture. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2017. A stylometric inquiry into hyperpartisan and fake news. *arXiv preprint arXiv:1702.05638*.

Victoria Rubin, Niall Conroy, Yimin Chen, and Sarah Cornwell. 2016. Fake news or truth? using satirical cues to detect potentially misleading news. In *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*, pages 7–17.

Prasha Shrestha, Sebastian Sierra, Fabio Gonzalez, Manuel Montes, Paolo Rosso, and Thamar Solorio. 2017. Convolutional neural networks for authorship attribution of short texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 669–674.

William Yang Wang. 2017. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*.

TakeLab at SemEval-2019 Task 4: Hyperpartisan News Detection

Niko Palić, Juraj Vladika, Dominik Čubelić, Ivan Lovrenčić, Maja Buljan, Jan Šnajder
Text Analysis and Knowledge Engineering Lab
Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, 10000 Zagreb, Croatia
{name.surname}@fer.hr
borat-sagdiyev team

Abstract

In this paper, we demonstrate the system built to solve the SemEval-2019 task 4: Hyperpartisan News Detection (Kiesel et al., 2019), the task of automatically determining whether an article is heavily biased towards one side of the political spectrum. Our system receives an article in its raw, textual form, analyzes it, and predicts with moderate accuracy whether the article is hyperpartisan. The learning model used was primarily trained on a manually prelabeled dataset containing news articles. The system relies on the previously constructed SVM model, available in the Python Scikit-Learn library. We ranked 6th in the competition of 42 teams with an accuracy of 79.1% (the winning team had 82.2%).

1 Introduction

The ability to quickly, precisely, and efficiently discern if a given article is hyperpartisan can prove to be beneficial in a multitude of different scenarios. Should we, for example, wish to evaluate if a certain news publisher delivers politically biased content, the best way to do so would be analyzing that very content. However, the sheer amount of articles modern news companies produce nowadays asks for an automated approach to the problem.

Spotting bias in text is both a well-known and challenging natural language problem. As bias can manifest itself in a covert or ambiguous manner, it is often hard even for an experienced reader to detect it. There was some research done on similar issues before (Doumit and Minai, 2011), but none specifically on the subject of hyperpartisan news.

The system described in this paper was built for Task 4 of the SemEval-2019 competition. The goal of the system, as set by the task, is to predict, as accurately as possible, whether a given article is hyperpartisan. While there were other criteria for evaluating the performance of the model (precision, recall, F1), we decided to optimize the program for the accuracy criterion, as the rankings were based solely on this measure. Accuracy, in this context, indicates the ratio of correctly predicted articles to the total number of articles. The final model reached an accuracy of 79.1%, which presents a decent score, considering the complexity of the problem and the available technology.

The system we built is based on the SVM model publicly available in Python's SciKit-Learn library (Pedregosa et al., 2011). Our model was trained on a handful of carefully chosen features derived from the given dataset and our understanding of the nature of bias. The dataset was split into a high-quality, manually labelled set of articles, and a large, but sub-par set of automatically labelled articles. By experimenting with the datasets, models, and features, we managed to create a system which ranked 6th in this competition.

2 Dataset Description

The dataset, which was provided by the task organizers, was divided into two separate clusters. The first and larger cluster consisted of one million news articles labelled solely by the political affiliation of the *publisher*. The second and much smaller cluster consisted of one thousand news articles labelled by people who read and evaluated them.

There was a substantial difference in labelling between these two datasets, so the quality and accuracy of the smaller dataset greatly overshadowed the abundance of articles in the larger dataset. Furthermore, the articles mostly came from large U.S. news publishers, such as: Fox News, CNN, Vox, etc. This predominance of prominent U.S. news networks and the substantial difference in quality largely impacted our feature design and, ultimately, our model selection.

The task itself was not divided into subtasks, but the submission on these two different datasets was regarded as a different subtask. The final test set, on which the main leaderboard is made, consisted solely of the articles from the smaller and more accurate dataset.

3 Model Description

Model selection and feature design were vastly influenced by the radical difference in quality between the two given datasets. As we mentioned, the larger, publisher-based dataset had a high number of mislabelled articles. For example, articles about diet tips, weather forecasts, or some other non-political news were often labelled as hyperpartisan news, solely on the fact that the publisher is classified as a generally biased news source. This mislabelling often caused our models to wrongly guess on what really indicates hyperpartisanism in news articles. Since the larger dataset often gave us relatively low accuracy, we decided to try a different approach.

Our first submitted model was trained only on the smaller and more accurate dataset. We decided to use SVC with GridSearch to maximize our accuracy on such a small dataset. Our best accuracy on the validation set, after cross-validation and with all features in place, was 77.9%. Furthermore, for our second submitted model, we decided to use a self-learning method to acquire more quality data from the larger dataset. Our first step was to train a logistic regression model on one-half of the smaller dataset, and, once trained, we tested that model on the larger dataset. All correctly classified articles were added to a new dataset. Once we extracted and combined all high-quality articles, we again used the SVC model. Our final accuracy on this model was also 77%.

4 Features

Our main tool used for processing news articles was word2vec[1] (Mikolov et al., 2013). Each word was converted into a vector, and all the word vectors generated from an article were then summed up. To prepare the dataset for word2vec, we used stemming and lemmatization tools from NLTK toolkit (Bird et al., 2009), and eliminated standard English stopwords. Also, in our earlier stages of development, we used chunking to get more meaningful information from the text.

[1]Additionaly, we experimented with GloVe (Pennington et al., 2014) and fastText (Bojanowski et al., 2016), but both models were outperformed by word2vec.

We ultimately ended up with a 300-dimensional vector for every article. This was the result of passing the preprocessed text into word2vec. Below, we elaborate on some other features we introduced to our model.

Publication date

Hyperpartisanism, or extreme bias, is a classification closely related to politics. As we were dealing predominatly with U.S. news outlets, we reasoned that news articles occurring around certain dates could demonstrate more hyperpartisanism. For example, months leading up to and following annual U.S. elections could be a mild indication of hyperpartisanism. Our first intuition was to only include months as a feature, since, in the U.S., certain election processes take place yearly. With further tests, we found that, if the year was also included, the accuracy improved by over 2%. That improvement could mean not only that elections produce more hyperpartisan news, but the type of the election, and maybe even the winner, could have an impact on news bias.

Website referencing

After inspecting a number of news articles in the dataset, we found that the majority of articles reference news sites whose objective political affiliation may be easily determined. Using that fact, we made a list of all known, extremely biased, U.S.-based news sources, and a list of objectively neutral news sources to counter the effect. By simply providing the number of extremely biased and neutral references, we improved the accuracy of our models by 2.3%. This could confirm that it is generally more common for news sources to reference other news sources with whom they share a similar political affiliation.

Sentiment analysis

Our initial assumption was that hyperpartisan articles tend to be more negative and aggressive than non-hyperpartisan articles. We used the sentiment intensity analyzer found in NLTK's sentiment library. The analyzer provides the scores for positivity, negativity, neutrality, and a compound score (i.e. aggregated score). We included these four as features in our model, which proved our theory correct and increased our accuracy results by 1.5%.

Model	Accuracy
Word2vec	0.58370
with added sentiment factor	0.58589
with added quotation counter	0.59874
with added date (month)	0.61360
with added date (month and year)	0.61782
with added NER counter	0.62556

Table 1: Validation results for our first SVC model on the by-publisher dataset, with particular features added one by one.

Model	Accuracy
Word2vec	0.75663
with added sentiment factor	0.76128
with added date (month and year)	0.76285
with added quotation counter	0.76904
with added trigger word counter	0.77981

Table 2: Validation results for our first SVC model on the by-article dataset, with particular features added one by one.

Trigger words
Analyzing the articles, we noticed that the writers of extremely biased news articles were prone to using *trigger words* more often than the writers of neutral news articles. With that in mind, we assembled a list of possible trigger words (containing mostly profanities). For each article, we took the count of *trigger words* in the text, and used it as input in the feature vector.

Named entity recognition
A named entity is a real-world object, such as a person, location, organization, product, etc., which can be denoted by a proper name. We decided to count, and use as another feature, the number politics-related named entities found in the articles. We assumed that the more named entities were found, the more biased an article would be. We used the software library spaCy[2] (Honnibal and Montani, 2017) and its named entity recognition tagger to extract mentions of various named entities, such as organizations, people, locations, dates, percentages, time, and money. We used the counts as inputs in the feature vector. Although counting most of the entity types dropped the total accuracy of the model, one type in particular was beneficial. It is a type called NORP and it denotes nationalities, religious groups, and political groups. Counting only the number of these named entities as a feature increased the model's accuracy slightly, by about 1%.

[2]https://spacy.io

5 Evaluation

First, we trained our model with the much larger but subpar *by-publisher* dataset, and then fine-tuned it using the much smaller but more precise *by-article* dataset.

5.1 Larger Dataset

Labels of the *by-publisher* dataset weren't as precise as the smaller dataset, which is why the accuracy results were lower, in the $58-62\%$ range. Table 1 showcases the results. We can see that adding a sentiment factor as a feature didn't change the overall accuracy for this dataset, but we should mention that it did increase the smaller dataset's accuracy much more. Adding the number of occurrences of biased publisher names quoted in the article increased the accuracy by about 1.3%. The largest increase in accuracy came with adding the date as a feature. Adding only the month as a feature increased the accuracy by 1.5%, but adding both month and year of article publication saw an additional increase of about 0.5%. Adding a counter of named entities (in particular: nationalities, religious groups, and political groups) increased the overall accuracy by just under 1%.

5.2 Smaller Dataset

Finally, the model with all of the features explained above was trained on the *by-article* dataset. While validating our results, the dataset was divided into five parts. 4⁄5 were used for training, and the remaining 1⁄5 was used for validation. The datasets were permutated, and, in the end, we took the average of the five permutations. We trained the model using 10 different classifiers. The validation results are shown in Table 3.

6 Conclusion

We described our system for hyperpartisan detection, developed for SemEval 2019 - Task 4. The essence of our system was SVC with GridSearch built on a variety of hand-crafted features. The task itself was a complex NLP problem, as hyperpartisanism detection in text often presents a problem even for an experienced human reader. Our main mission was to extract a few quality features that would help us tackle this convoluted problem. Future work includes experiments with different

Classifier	Accuracy
Logistic Regression	0.70246
SVC	0.76590
SVC GridSearch	**0.77981**
GaussianNB	0.68344
RandomForestClassifier	0.71649
MLPClassifier	0.76117
AdaBoostClassifier	0.70866
LinearSVC	0.69619
GradientBoostingClassifier	0.72242
IsolationForest	0.35153

Table 3: Validation results for the final model on the by-article dataset. Classifier used for submission is in boldface.

Team name	Accuracy
bertha-von-suttner	0.822
vernon-fenwick	0.820
sally-smedley	0.809
tom-jumbo-grumbo	0.806
dick-preston	0.803
borat-sagdiyev	**0.791**
morbo	0.790
howard-beale	0.783
ned-leeds	0.775
clint-buchanan	0.771

Table 4: Final rankings on the main task. Our submission is in boldface.

classifiers, and possibly neural networks. Furthermore, our mission in feature extraction will continue and we hope to find more linguistic features that would help us to improve and upgrade our model.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606.*

Sarjoun Doumit and Ali Minai. 2011. Online news media bias analysis using an lda-nlp approach. In *International Conference on Complex Systems.*

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear.*

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019).* Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781.*

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Team Fernando-Pessa at SemEval-2019 Task 4: Back to Basics in Hyperpartisan News Detection

André Ferreira Cruz
Universidade do Porto
Faculdade de Engenharia
andre.ferreira.cruz@fe.up.pt

Gil Rocha
Universidade do Porto
Faculdade de Engenharia
LIACC
gil.rocha@fe.up.pt

Rui Sousa-Silva
Universidade do Porto
Faculdade de Letras
CLUP
rssilva@letras.up.pt

Henrique Lopes Cardoso
Universidade do Porto
Faculdade de Engenharia
LIACC
hlc@fe.up.pt

Abstract

This paper describes our submission[1] to the SemEval 2019 Hyperpartisan News Detection task. Our system aims for a linguistics-based document classification from a minimal set of interpretable features, while maintaining good performance. To this goal, we follow a feature-based approach and perform several experiments with different machine learning classifiers. On the main task, our model achieved an accuracy of 71.7%, which was improved after the task's end to 72.9%. We also participate in the meta-learning sub-task, for classifying documents with the binary classifications of all submitted systems as input, achieving an accuracy of 89.9%.

1 Introduction

Hyperpartisan news detection consists in identifying news that exhibit extreme bias towards a single side (Potthast et al., 2018). The shift, in news consumption behavior, from traditional outlets to social media platforms has been accompanied by a surge of fake and/or hyperpartisan news articles in recent years (Gottfried and Shearer, 2017), raising concerns in both researchers and the general public. As ideologically aligned humans prefer to believe in ideologically aligned news (Allcott and Gentzkow, 2017), these tend to be shared more often and, thus, spread at a fast and unchecked pace. Moreover, there is a large intersection of 'fake' and 'hyperpartisan' news, as 97% of fake news articles in BuzzFeed's Facebook fact-check dataset are also hyperpartisan (Silverman et al., 2016).

However, the detection/classification and consequent regulation of online content must be done with careful consideration, as any automatic system risks unintended censorship (Akdeniz, 2010). As such, we aim for a linguistically-guided model from a set of interpretable features, together with classifiers that facilitate inspection of what the model has learned, such as Random Forests (Ho, 1995), Support Vector Machines (Cortes and Vapnik, 1995) and Gradient Boosted Trees (Drucker and Cortes, 1996). Neural network models are left out for their typically less self-explanatory nature.

The SemEval 2019 Task 4 (Kiesel et al., 2019) challenged participants to build a system for hyperpartisan news detection. The provided dataset consists of 645 manually annotated articles (*by-article dataset*), as well as 750,000 articles automatically annotated publisher-wise (*by-publisher* dataset, split 80% for training and 20% for validation). Systems are ranked by accuracy on a set of unpublished test articles (from the *by-article* dataset), which has no publishers in common with the provided train dataset, preventing accuracy gains by profiling publishers. All experiments on this paper are performed on the *gold-standard* (*by-article*) corpus, as this was the official dataset.

The rest of the paper is organized as follows. Section 2 describes our pre-processing, feature selection, and the system's architecture. Section 3 analyzes our model's performance, evaluates each feature importance, and goes in-depth on some classification examples. Finally, Section 5 draws conclusions and sketches future work.

2 System Description

We propose a feature-based approach and experiment with several machine learning algorithms, namely support vector machines with linear ker-

[1] https://github.com/AndreFCruz/semeval2019-hyperpartisan-news

nels (SVM), random forests (RF), and gradient boosted trees (GBT). Our submission to the task was a RF classifier, as this was the best performing at the time. However, after the task's end we found a combination of hyperparameters that turned GBT into the best-performer. We detail all results in the following sub-sections.

All classifiers were implemented using *scikit-learn* (Pedregosa et al., 2011) for the *Python* programming language, and all were trained on the same dataset of *featurized* documents. In this section we describe the data pre-processing, our selection of features, as well as the classifiers' grid-searched hyperparameters.

2.1 Feature Selection

The statistical analysis of natural language has been widely used for stylometric purposes, in particular in order to define linguistic features to measure author style. These include, among others: document length, sentence and word length, use of punctuation, use of capital letters, and frequency of word n-grams; type-token ratio (Johnson, 1944); and frequency of word n-grams (see e.g. Stamatatos (2009) for a thorough survey of authorship attribution methods). Although these features have been successfully used in authorship attribution to establish the most likely writer of a target text among a range of possible writers (Sousa-Silva et al., 2010, 2011), research on how these features can be used to analyze group authorship – and subsequently identify an ideological slant – is less demonstrated. Therefore, we build upon previous research on Computer-Mediated Discourse Analysis (Herring, 2004) to test the use of these features to detect hyperpartisan news.

We compute a minimal set of style and complexity features, partially inspired by Horne and Adali (2017), as well as a bag of word n-grams. For tokenization we use the Python Natural Language Toolkit (Bird et al., 2009).

Our features are as follow: *num_sentences* (number of sentences in the document); *avg_sent_word_len* (average word-length of sentences); *avg_sent_char_len* (average character-length of sentences); *var_sent_char_len* (variance of character-length of sentences); *avg_word_len* (average character-length of words); *var_word_len* (variance of character-length of words); *punct_freq* (relative frequency of punctuation characters); *capital_freq* (relative frequency of capital letters); *types_atoms_ratio* (type-token ratio, a measure of vocabulary diversity and richness); and frequency of the k most frequent word n-grams.

Regarding word n-grams, we use $k = 50$ and $n \in [1, 2]$, as we empirically found these values to perform well while maintaining a small feature set. Moreover, we ignore n-grams whose document frequency is greater than 95%, as well as 1-grams from a set of known English stop-words (from *scikit-learn*'s stop-word list), whose frequency we assume to be too high to be distinctive. Text tokens and stop words are *stemmed* using the Porter stemming algorithm (Porter, 1980).

2.2 Hyperparameters

For RF, we use the following hyperparameter values: 100 estimators; minimum samples at leaf = 1; criterion = gini; minimum samples to split = 2.

For GBT, we use the following hyperparameter values: 50 estimators; minimum samples at leaf = 3; loss = exponential; learning rate = 0.3; minimum samples to split = 5; max depth = 8.

For SVM model, we use the following hyperparameter values: penalty parameter C = 0.9; penalty = l2; loss function = squared hinge.

These hyperparameter values are the result of extensive grid searching for each model, selecting the best performing models on 10-fold cross-validated results.

3 Results and Discussion

Table 1 shows the results of the models over 10-fold cross validation (top rows), and on the official test set (bottom rows). Besides our models, we show the performance of the provided baseline as well as the best performing submission to the task (last row). As results on the official test set were hidden during the duration of the task, we used cross-validated results to guide our decision-making in improving the models.

3.1 Feature Analysis

Making use of our choice of classifiers, we are able to interpret and analyze the most important features, as well as trace back the decision path for every document along each of the ensemble's estimators (RF and GBT).

Figures 1 and 2 show measures of feature importance for the RF and GBT models. Figure 1 shows the top features by *mean impurity decrease*

Dataset	*Model*	*Accuracy*	*Precision*	*Recall*	*F1*
10-fold CV on *by-article-training*	GBT	75.9	71.4	59.4	64.6
	RF	76.3	74.6	55.4	63.3
	SVM	72.7	71.3	45.5	55.1
by-article-test (official)	GBT	72.9	78.1	63.7	70.2
	RF	**71.7**	**80.6**	**57.0**	**66.8**
	Baseline	46.2	46.0	44.3	45.1
	Best Team	82.2	87.1	75.5	80.9

Table 1: Models performance: values in the top rows result from 10-fold cross-validation on the *by-article-training* set, and values in the bottom rows report evaluation on the official test set through TIRA (Potthast et al., 2019). RF refers to our task submission, while GBT is our best performing model, submitted after the task's closing.

on a feature's nodes, averaged across the ensemble's estimators/trees and weighted by the proportion of samples reaching those nodes (Breiman, 2001). Similarly, Figure 2 shows the top features by *relative accuracy decrease* (averaged across the ensemble's estimators) as the values of each feature are randomly permuted (Breiman, 2001).

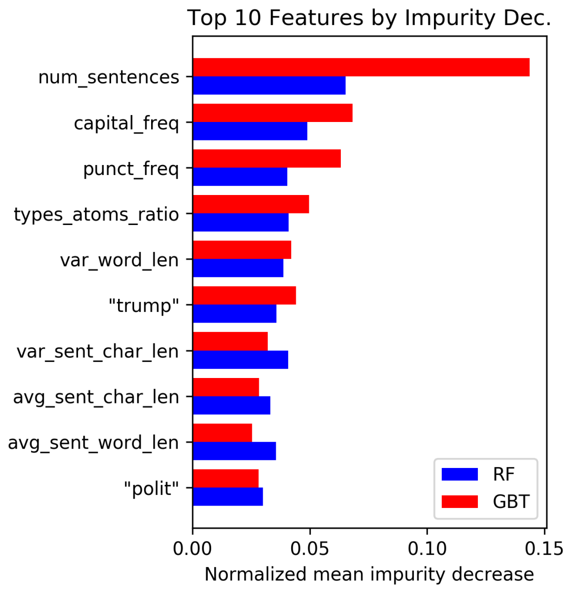


Figure 1: Top features by mean *impurity* decrease, sorted by average value among the two classifiers.

Interesting properties emerge from analyzing feature importance, notably that the *number of sentences* and the *frequency of capital letters* are the most important features on both measures. Moreover, the RF model tends to have a longer-tailed distribution of feature importances, while the GBT model tends to focus on a smaller subset of features for classification.

Interestingly, two 1-grams make it into the top-10 features by impurity decrease: 'trump' and 'polit'. Reliance on n-grams could present a larger problem, as these may refer to entities with a high variance of media attention. For instance, words like 'Hillary' or 'Obama' (which appear in the top-20 features) are probably not seen as often nowadays as they were back in 2016. As such, we are confident in the generalization capacity of the models, as the most discriminative features are mostly style and language-complexity features, which do not suffer from the previously stated biases of n-grams.

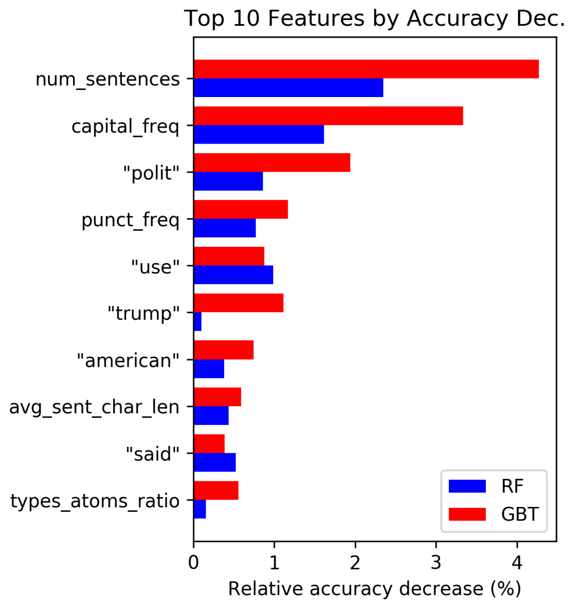


Figure 2: Top features by relative *accuracy* decrease, sorted by average value among the two classifiers.

3.2 Analysis of Predictions

In order to better understand our model's decision making, we analyze differences in distributions of document features for each predicted class, and compare them with the gold-standard values.

As seen in Table 2, articles predicted as hyperpartisan have a higher number of sentences, but each with lower length than mainstream articles, and with decreased vocabulary diversity (smaller type-token ratio). The frequency of the word 'trump' is also noticeably higher in hyperpartisan articles. There is a good alignment of *predicted* and *gold* articles, when projected onto this feature space.

Feature	Pred. Avg.		Gold Avg.	
	H	M	H	M
num_sentences	39.7	19.4	37.5	20.4
capital_freq	0.046	0.058	0.046	0.058
punct_freq	0.030	0.032	0.030	0.033
type_atoms_ratio	0.54	0.60	0.55	0.59
var_word_len	2.70^2	2.71^2	2.69^2	2.71^2
"trump"	6.13	2.54	5.86	2.64
var_sen_char_len	91.5^2	162^2	93.5^2	162^2
avg_sen_char_len	127	156	129	156
avg_sen_word_len	24.9	29.2	25.1	29.1
"polit"	1.37	0.35	1.34	0.35

Table 2: Average values for articles *predicted* (Pred.) hyperpartisan (H) or mainstream (M), and for their *ground-truth* (Gold), for top-10 features by impurity decrease.

4 Meta-learning Task

After the main task's end, organizers challenged participants to compete on a meta-learning task. This task's dataset consisted of the predictions made by each of the 42 submitted systems on the same test-set articles. Notably, a simple majority vote classifier (with the predictions of all 42 systems as input) achieved accuracy of 88.5%, substantially better than the best performing system's accuracy of 82.2%.

While a voting classifier performed considerably well, we intuitively postulated that the votes of the best-n classifiers (accuracy-wise) would perform better. Figure 3 shows the accuracy of n majority vote classifiers, from the top-*42* systems to the top-*1* system. The best performance is achieved using the top-*12* classifiers. However, in Figure 3, we can observe fluctuations in performance while removing the worst classifiers. This means that combining worst classifiers as we do in this task can yield performance improvements. We conclude that there is no discernible correlation between performance and smaller n. We leave as future work further investigation on what characteristics of the classifiers contribute to the fluctuations of the overall performance.

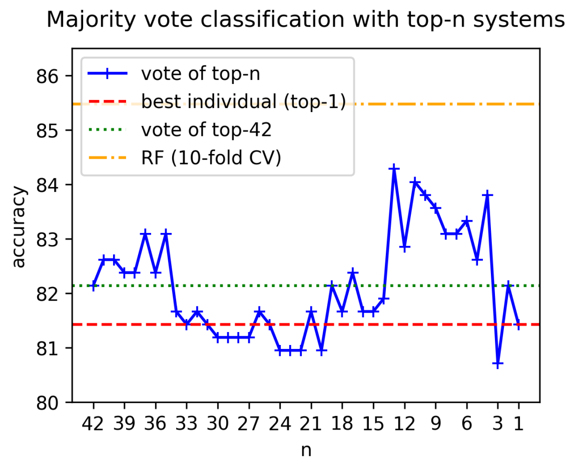


Figure 3: Performance of a majority vote classifier using the top-n best performing systems (by accuracy), on the provided *by-article-meta-training* dataset.

Our final submission for this sub-task consisted of a Random Forest model, whose features were the predictions of all 42 submitted systems, as well as an extra column with the average vote of all systems. See Table 3 for the final performance on the official *by-article-meta-test* dataset.

Model	*A*	*P*	*R*	*F1*
RF	89.9	89.5	90.4	90.0
Majority Vote (42)	88.5	89.2	87.5	88.3
Baseline	52.9	52.5	59.6	55.9

Table 3: Performance on the meta-learning task, evaluated on the *by-article-meta-test* dataset through TIRA.

5 Conclusions

We experimented with several models for hyperpartisan news detection, supplied with a small set of 9 linguistically-inspired features in addition to the 50 most frequent n-grams. Our official submission is a Random Forest model, which achieved an accuracy of 71.7%. On the meta-learning sub-task we achieved an accuracy of 89.9%.

For future work, we intend to further explore differences in writing style between hyperpartisan and mainstream articles, as well as ensembles of individually distinct classifiers, as it seems a promising path towards more accurate hyperpartisan news detection.

Acknowledgments

André Ferreira Cruz is supported by the Calouste Gulbenkian Foundation, under grant number 226338. Gil Rocha is supported by a PhD studentship (with reference SFRH/BD/140125/2018) from Fundação para a Ciência e a Tecnologia (FCT). This research is partially supported by project DARGMINTS (POCI/01/0145/FEDER/031460), funded by FCT.

References

Yaman Akdeniz. 2010. To block or not to block: European approaches to content regulation, and implications for freedom of expression. *Computer Law & Security Review*, 26(3):260–272.

Hunt Allcott and Matthew Gentzkow. 2017. Social media and fake news in the 2016 election. *Journal of economic perspectives*, 31(2):211–36.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Harris Drucker and Corinna Cortes. 1996. Boosting decision trees. In *Advances in neural information processing systems*, pages 479–485.

Jeffrey Gottfried and Elisa Shearer. 2017. Americans online news use is closing in on tv news use. *Pew Research Center*.

Susan C Herring. 2004. Computer-Mediated Discourse Analysis: An Approach to Researching Online Behavior. In S. A. Barab, R. Kling, and J. H. Gray, editors, *Designing for Virtual Communities in the Service of Learning*, pages 338–376. Cambridge University Press, Cambridge.

Tin Kam Ho. 1995. Random decision forests. In *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282. IEEE.

Benjamin D Horne and Sibel Adali. 2017. This just in: fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. In *Eleventh International AAAI Conference on Web and Social Media*.

Wendell Johnson. 1944. Studies in language behavior: A program of research. *Psychological Monographs*, 56(2):1–15.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Martin F Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. TIRA Integrated Research Architecture. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 231–240. Association for Computational Linguistics.

Craig Silverman, Lauren Strapagiel, Hamza Shaban, Ellie Hall, and Jeremy Singer-Vine. 2016. Hyperpartisan facebook pages are publishing false and misleading information at an alarming rate. *Buzzfeed News*.

Rui Sousa-Silva, Gustavo Laboreiro, Luís Sarmento, Tim Grant, Eugénio Oliveira, and Belinda Maia. 2011. twazn me!!! ;(' Automatic Authorship Analysis of Micro-Blogging Messages. In *Lecture Notes in Computer Science 6716 Springer 2011*, volume Natural La, pages 161–168, Berlin and Heidelberg. Springer – Verlag.

Rui Sousa-Silva, Luís Sarmento, Tim Grant, Eugénio C Oliveira, and Belinda Maia. 2010. Comparing Sentence-Level Features for Authorship Analysis in Portuguese. In *Computational Processing of the Portuguese Language*, pages 51–54.

Efstathios Stamatatos. 2009. A Survey of Modern Authorship Attribution Methods. *Journal of the American Society for Information Science and Technology*, 60(3):538–556.

Team Harry Friberg at SemEval-2019 Task 4: Identifying Hyperpartisan News through Editorially Defined Metatopics

Nazanin Afsarmanesh, Jussi Karlgren, Peter Sumbler, Nina Viereckel
Gavagai
Stockholm , Sweden
{nazanin, jussi, peter, nina}@gavagai.io

Abstract

This report describes the starting point for a simple rule based hypothesis testing excercise on identifying hyperpartisan news items carried out by the Harry Friberg team from Gavagai. We used manually crafted *metatopics*, topics which often appear in hyperpartisan texts as rant conduits, together with tonality analysis to identify general characteristics of hyperpartisan news items. While the precision of the resulting effort is less than stellar— our contribution ranked 37th of the 42 successfully submitted experiments with overly high recall (95%) and low precision (54%)—we believe we have a model which allows us to continue exploring the underlying features of what the subgenre of hyperpartisan news items is characterised by.

1 Hyperpartisanism

Hyperpartisan news are news items that are strongly argumentative and one-sided. However, being biased is not enough to be characterised as being hyperpartisan, neither is it enough for a news item to use strong language. Confounders for this task includes items that use strong language without being partisan, items that are subjective but not ”hyper”, and items that report dispassionately on typically hyperpartisan topics.

We hypothesise that authors of hyperpartisan texts are in the process of performing a sub-genre of their own, intended not as much to convey the reader information about some state of the world but to mobilise sentiment and affect in the readership, and establishing a shared attitudinal space. Taking this point of departure, we assume that the linguistic items employed by the authors of hyperpartisan text are not only related to the topics under discussions, nor to argumentation, but also include some genre-specific features to explicitly signal hyperpartisanness. This report describes an experiment based on these starting points, performed on data from the 2019 SemEval task on Hyperpartisan News Detection. (Kiesel et al., 2019)

2 Gavagai Explorer

The Gavagai Explorer is a commercially available tool which provides an end-to-end solution for the analysis of unstructured text data (Espinoza et al., 2018). We have in these experiments made use of its components for topic clustering, sentiment analysis, and concept modelling.

2.1 Trigger Topics

The *topic clustering* is based on lexical cues, and can be used to detect what themes and topics are prevalent in some set of e.g. customer feedback messages. Here, we used the topic clustering to establish what sort of themes were frequent in the hyperpartisan training set.

We postulate that many *metatopics* turn out to become lightning rods for hyperpartisan argumentation, somewhat (but not entirely) unpredictably. A characteristic of some of the more extreme sample items was that a hyperpartisan rant will bring in additional only marginally related topics into an argumentation. We identified a small set of potential rant metatopics using the topic clustering mechanism in the Gavagai Explorer. A breakdown of these topics with some example terms can be found in Table 1.

2.2 Trigger Attitudes

The *concept modeling* tool allows an analyst to define measures based on lexical items. *Sentiments* are a special case, applied to the palette of human emotion. On entering some seed words, the user is presented with semantically similar terms acquired from a distributional model (Sahlgren et al., 2016). The user accepts terms which are relevant,

Topic	Example terms
American	across america, america first, god bless america, ...
Elites	elite, establishment, oligarchy, ...
Freedom of Speech	first amendment rights, freedom of speech, press freedom, ...
Islam	islamist, islamism, muslim, ...
Media	cable news, mainstream media, twitter, ...
People	people, these people, the people, ...
Political Movements	alt-right, bolshevik, marxist
Politicians	party leaders, political party, politician, ...
Populism	populism, populist, ...
Public Safety	felon, incriminating, predator, ...
Race	black lives matter, black people, white people, ...
Sexual Rights	reproductive rights, same sex marriage, transgender, ...
Support	support, supportive, supported, ...
Woman	woman, women, ...

Table 1: Trigger topics and some of terms that indicate them. The full list with all terms is available at https://www.gavagai.io/blog/2019/06/06/gavagai-identifying-hyperpartisan-news/ .

Concept	Example terms
Certainty	blatantly, undeniably, hands-down, ...
Cynical	bizarre, far-fetched, ludicrous, ...
Exasperation	and once again, for some reason, yet again, ...
Failure	catastrophe, complete failure, disaster, ...
Nonsense	arrogant, babble, claptrap, ...
Puffery	landmark, pioneering, visionary, ...
Weasel Words	recent study, widely acknowledged, been claimed, ...
Win and Lose	bruised, humiliated, vanquished, ...

Table 2: Trigger attitudes and some of terms that indicate them. The full list with all terms is available at https://www.gavagai.io/blog/2019/06/06/gavagai-identifying-hyperpartisan-news/ .

which are in turn used to provide more suggestions in the following iteration. Here, we used the concept modelling tool to define *trigger attitudes* such as those shown in Table 2.

We find that strongly expressed attitudes not necessarily mean that an article is hyperpartisan, but that the combination of a trigger topic together with negative sentiment appears to be indicative of hyperpartisanism. The sentiment analysis component identifies several types of polar language, and measures the intensity of expression in each item using both presence of polar terms and of amplifier terms such as ”extremely” and ”very”. In addition to standard polar sentiments we used the concept modeling tool to build a set of concepts tailored to observable presence in hyperpartisan texts (Karlgren et al., 2012).

2.3 Trigger Styles

Besides topical specificity we expect hyperpartisan texts to be couched in specific styles, as already established in previous studies (Potthast et al., 2018). We compared some stylistic features known to us to have discriminative power in other contexts, such as counts of exclamation marks, question marks, digits, capital letters, capitalised words, type token ratio, word length, sentence length etc. We found that the strongest single stylistic feature was the presence of many exclamation marks, in conjunction with trigger topics, while most other features on their own were less indicative. This is an indication that the authors of hyperpartisan texts appear to adhere to most stylistic conventions of the news genre.

3 Rule Based Fusion

We combined the above evidence in a rule based model, to achieve reasonably high explanatory power of results for downstream application. Through analysis of the training data, we distilled the results into the following pieces of reasoning, applied in the order given here:

1. Presence of many trigger topics (> 3) in an article, indicates it is hyperpartisan.

2. Presence of at least one trigger topic and a negative sentiment score for an article indicates it is hyperpartisan.

Figure 1: K W Gullers and Stieg Trenter, 1950s

3. A positive sentiment score combined with lack of trigger topics indicates a non-hyperpartisan article.

4. Presence of at least one trigger topic together with a high type token ratio *or* high ratio of questions in an article indicates it is hyperpartisan.

5. A high trigger attitude score (given in Table 2) indicates an article is hyperpartisan.

4 Results

The end results of our experiment on the by-article test set were decidedly underwhelming, with our contribution ranked 37th of 42 experiments. Our combined experimental pipeline yielded high recall (95%) and low precision (54%), meaning that it turned out to be overly sensitive to the features it was trained on. The rule set given above triggered for too many non-hyperpartisan items, with the last rule being the most permissive. We still believe that informed and hypothesis-driven analysis of content, rather than an end-to-end learning models, will result in a model of greater generality and greater explanatory power, but that the rule based combination should have been done using some learning scheme. While the precision of the resulting effort is less than stellar, we believe we have a model which allows us to continue exploring the underlying features of what the sub-genre of hyperpartisan news items is characterised by, and we also believe that the explicit representation of what features are in play will afford end users greater trust in the system's classification results.

5 Namesake

Harry Friberg was a fictional photojournalist and the protagonist of a series of crime novels by Stieg Trenter (1914-1967). The character first appeared in the novel *Farlig fåfänga*, 1944, and continued in a series of novels which have since become popular classics for their depiction of Stockholm in the 1950s. Harry Friberg was modeled on the internationally recognised photojournalist K W Gullers (1916—1998), a friend of the author.

References

Fredrik Espinoza, Ola Hamfors, Jussi Karlgren, Fredrik Olsson, Per Persson, Lars Hamberg, and Magnus Sahlgren. 2018. Analysis of open answers to survey questions through interactive clustering and theme extraction. In *Proceedings of the 2018 Conference on Human Information Interaction&Retrieval*, pages 317–320. ACM.

Jussi Karlgren, Magnus Sahlgren, Fredrik Olsson, Fredrik Espinoza, and Ola Hamfors. 2012. Usefulness of sentiment analysis. In *European Conference on Information Retrieval*, pages 426–435. Springer.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 231–240. Association for Computational Linguistics.

Magnus Sahlgren, Amaru Cuba Gyllensten, Fredrik Espinoza, Ola Hamfors, Jussi Karlgren, Fredrik Olsson, Per Persson, Akshay Viswanathan, and Anders Holst. 2016. The Gavagai living lexicon. In *Language Resources and Evaluation Conference*. ELRA.

Team Howard Beale at SemEval-2019 Task 4: Hyperpartisan News Detection with BERT

Osman Mutlu[*]
Koç University
İstanbul, Sarıyer
omutlu@ku.edu.tr

Ozan Arkan Can[*]
Koç University
İstanbul, Sarıyer
ocan13@ku.edu.tr

Erenay Dayanık
University of Stuttgart
Stuttgart
erenaydayanik@gmail.com

Abstract

This paper describes our system for SemEval-2019 Task 4: Hyperpartisan News Detection (Kiesel et al., 2019). We use pretrained BERT (Devlin et al., 2018) architecture and investigate the effect of different fine tuning regimes on the final classification task. We show that additional pretraining on news domain improves the performance on the Hyperpartisan News Detection task. Our system[1] ranked 8th out of 42 teams with 78.3% accuracy on the held-out test dataset.

1 Introduction

With the rapid spread of the Internet and next-generation media development, people started to follow news through the Internet by abandoning de facto sources such as television and radio. Recent studies reveal that 43% of Americans report often getting news online (Shearer and Gottfried, 2017). In parallel with that, there also has been a massive improvement in the NLP research in news domain to keep the content true, fair and unbiased. SemEval-2019 Task 4: Hyperpartisan News Detection, is yet another attempt under this objective. Hyperpartisan is defined as being extremely biased in favor of a political party (Bastos and Mercea, 2017) and the aim of the shared task is to detect hyperpartisan argumentation in news text. Though it is an important task by itself, hyperpartisan argument detection is also considered as a very first step (or even replacement) of fake news detection, because it has been shown by (Potthast et al., 2018) that there is a high positive correlation between having a hyperpartisan argumentation and being fake for news items.

In this shared task, we seek to model this problem as a text classification task. In general, the task aims to label the text in the question with one or more classes or categories. The main question of text classification is how to mathematically represent the words/tokens such that they retain their original meaning in the context they appear. This question has been tried to be answered in many different ways so far. In earlier work, people mainly used the ”bag of words” approach in algorithms such as Naive Bayes, Decision Tree, and SVM. Then, (Mikolov et al., 2013) advanced the field further by introducing word embeddings, capturing a somewhat meaningful representation of words. However, recent studies (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2018) showed that contextual word embeddings perform quite better than traditional word embeddings in many different NLP tasks as a result of their superior capacity of meaning representation. Among those, BERT attracts researchers most because of (i) its transformer based architecture enabling faster training and (ii) state of the art results in many different tasks.

Though it is quite new, BERT has been tried in many different domains than the one proposed in Devlin et al. (2018). However, almost all of these studies have two things in common: they don't start training BERT from scratch and the target domain contains very limited data (Zhu et al., 2018; Yang et al., 2019; Alberti et al., 2019). In this study, on the other hand, we address (1) the performance of BERT by comparing its domain specific pre-trained and fine-tuned performances, and (2) in the setting where the target domain has extensively more data. In the following sections, we first summarize the BERT architecture, then give details of shared task data set, and then describe experimental setups we used to train BERT model. In the results section, we compare the performance of BERT under different settings and share our submission results for the shared task.

[*]equal contribution
[1]https://github.com/ozanarkancan/hyperpartisan

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 1007–1011
Minneapolis, Minnesota, USA, June 6–7, 2019. ©2019 Association for Computational Linguistics

2 Method

Transformer[2] (Vaswani et al., 2017) originally came out as a machine translation architecture and it uses the idea of self attention mechanism (Parikh et al., 2016; Lin et al., 2017). It has an encoder-decoder design and both parts use the same novel multi-head attention mechanism. The encoder part takes an input sentence and derives a representation from it using this attention mechanism. Afterwards, the decoder generates the target sentence by performing multi-headed attention over the encoder stack.

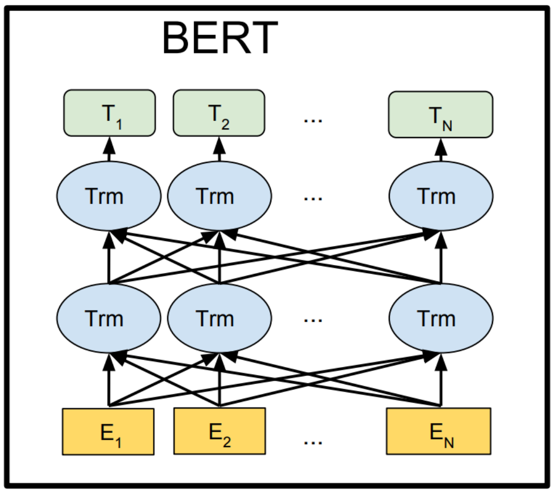


Figure 1: BERT Architecture (Devlin et al., 2018).

Figure 1 illustrates the architecture of the model. BERT learns bidirectional representations jointly on both left and right context of text making use of the encoder part of the Transformer. Devlin et al. (2018) introduced two unsupervised tasks to pretrain this architecture, Next Sentence Prediction and Masked Language Modeling. In Next Sentence Prediction task, the goal is to determine whether the sentence comes after the specified previous sentence or not. It takes two sentences as input, the latter being in its original form 50% of the time, while other times it can be any random sentence from the corpus. In Masked Language Modeling task, 15% of the words in the input sentences are masked and the model tries to predict these words. Training takes place with the combined loss of these two unsupervised tasks. Resulting representations can be further fine-tuned with a task specific layer on the top for a number of NLP tasks using appropriate supervised data.

[2]http://nlp.seas.harvard.edu/2018/04/03/attention.html

In this study, we use an open source PyTorch implementation[3] of BERT architecture. We make use of BERT-Base pretrained model provided by Devlin et al. (2018) in order to avoid pretraining from scratch. Similar to Devlin et al. (2018), we use the representation obtained from the last layer for the first token (i.e. ”[CLS]”) for the sentence representation and a softmax classifier on top of it for predicting hyperpartisanship.

3 Experiments

In this section, we first introduce data provided by the shared task and the data preprocessing step. Then, we give the details of our experiments and results with BERT under pretraining and fine-tuning settings.

3.1 Data

Task provides data that consist of 750.000 articles labelled portal-wise and 645 articles labelled manually, and they divide the former into 600.000 and 150.000 as train and development set. Portal-wise data is labelled as hyperpartisan or not, according to publishers known affinities provided by BuzzFeed journalists or MediaBiasFactCheck.com. In our experiments, we first shuffled and then split the portal-wise data into three: 705.000, 40.000, 5.000 articles for train, development and test respectively.

3.2 Preprocessing

For all our experiments we remove some unwanted text from the articles. We replaced HTML character reference for ampersand and numeric entity reference, and removed adjacent underscore characters which is possibly used as a replacement for classified information in data. We also removed lines, solely containing ”*” characters, used for separation of different news in the same article.

3.3 Input Representation

BERT restricts the input length to a maximum of 512 tokens. We select the first n tokens from the beginning of the article, because using the lead sentences of a news article has been found to be more effective for some NLP tasks (Wasson, 1998). We use the same tokenization method and embeddings as Devlin et al. (2018) to represent the words.

[3]https://github.com/huggingface/pytorch-pretrained-BERT

3.4 Fine-tuning Only

In order to show how BERT performs in news domain, our first attempt was to use the training data to only fine-tune the pretrained model for classification. We used BERT-Base which consists of 12 transformer blocks on top of each other applying 12 headed attention mechanism, hidden size of 768 and a total of 110 million parameters. We set 16 as our batch size and 2e-5 as our learning rate as recommended by Devlin et al. (2018) for fine-tuning on classification tasks.

Max Length	**Dev**		**Test**	
	Accuracy	*F1*	*Accuracy*	*F1*
128	84.99	84.91	84.40	84.36
256	88.91	88.89	**88.31**	88.31
512	**89.12**	89.09	88.15	88.14

Table 1: Classification results on our portal-wise data splits with fine-tuned BERT.

We performed experiments using 128, 256 and 512 as our maximum sequence lengths and found out that 256 gives us the best test results, as shown in Table 1. Although the results for experiments with maximum sequence lengths of 256 and 512 are relatively close to each other, we chose 256 for computational efficiency. From these results, we can argue that for news articles, the first 128 tokens do not carry enough information.

3.5 Pretraining + Fine-tuning

For the pretraining step, the data used by two unsupervised tasks need to be generated. For the Next Sentence Prediction task, originally, one would go over the articles sentence by sentence to generate pretraining data, but our data is not made of split sentences. To avoid using a tool for sentence splitting, as it would take too much time in large scale, for each document from the training data, we extract a chunk of text with a random length sampled from a uniform distribution defined as an interval between %15 and %85 of the maximum sequence length. The reason for this is to make the model more robust to non-sentential input and leave space for the second sentence. As the second sentence, 50% of the time, we select the chunk following the original one with a length that is complementing the first chunk's length up to maximum sequence length. Other times, when we need the next sentence to be random, we take a random chunk from other documents. We extract more than one sample from a single document, avoiding overlapping between chunks. For Masked LM task, we follow the same approach with Devlin et al. (2018).

Model	**Combined Loss**
BERT-Base	3.65
Our Version	**1.79**

Table 2: Results on the held-out dataset for pretraining tasks.

At the end of pretraining data generation process, we accumulated near 3.5 million samples, only running the process once on our train split, so without any duplication unlike Devlin et al. (2018) because of time restrictions. We also generated a small held-out dataset using our test split to use in evaluation. Starting from the pretrained model of BERT-Base instead of a cold start, we trained the model with a learning rate of 3e-5 and 256 as the maximum sequence length for 290k iterations. Table 2 presents the combined loss of two unsupervised tasks on the held-out data for original BERT-Base and further pretrained model with the generated data. Results show that pretraining BERT further with data from an unseen domain greatly increases its representational power.

Model	**Dev**		**Test**	
	Accuracy	*F1*	*Accuracy*	*F1*
Fine-Tuning Only	88.91	88.89	88.31	88.31
Pretraining +Fine-Tuning	**89.69**	89.67	**89.30**	89.29

Table 3: Comparison of fine-tuning only and pretraining + fine-tuning models.

After this step, we applied the same fine-tuning as previous section with the same parameters. Table 3 demonstrates that pretraining BERT with domain specific data using unsupervised tasks improves the performance of the model on the supervised classificiation task.

4 Shared Task Results

The evaluation of SemEval-2019 Task 4, Hyperpartisan News Detection task is done through the online platform of TIRA (**?**). It serves as a means of blind evaluation of the submitted model. Accuracy is used as the official evaluation metric and the deciding test set is an another manually labelled news articles set named "by-article-test-set" which was kept hidden from the participants.

Model	article-test				publisher-test			
	Accuracy	*Precision*	*Recall*	*F1*	*Accuracy*	*Precision*	*Recall*	*F1*
Fine-Tuning (publisher) + Fine-Tuning (article)	**78.3**	83.71	70.38	76.5	63.45	67.98	50.85	58.18
Pretraining (publisher) + Fine-Tuning (publisher) + Fine-Tuning (article)	73.4	66.81	92.99	77.76	64.15	60.64	80.6	69.21
Pretraining (publisher) +Fine-Tuning (publisher)	60.82	57.11	86.94	68.93	**67.25**	62.45	86.5	72.53

Table 4: Shared task results.

In our first attempt, we fine-tuned BERT with portal-wise train split using development set to get the best model. After this we further train it with 645 manually labeled data (i.e. ”by-article-train-set”), because it comes from the same sample as test data.

In our last attempt, we pretrained BERT with our portal-wise train split, and then fine-tune it as described before. Again, we further fine-tune our model with ”by-article-train-set” data. The results of our two attempts can be seen in Table 4. The third model in the table is to show the effect of the last fine-tuning step on ”by-article-train-set”.

Looking at the results of second and third models on ”by-article-test-set” shows us, although we fine-tune BERT with supervised data for the same classification task, fine-tuning on ”by-article-train-set” improves the results drastically. This may be rooted from the domain difference in between ”by-article-test-set” and portal-wise train data.

Although our experiments (Table 3) show us that pretraining BERT further with data from news domain has a positive effect on overall accuracy, we are not able to observe the similar effect on ”by-article-test-set”. The second model adapts to the publisher domain more than the first model does because of the extensive pretraining before fine-tuning. As the difference between publisher and article is highly notable from the findings before, overfitting to the publisher domain might end up hurting the generalization of the model. So, this would explain the unexpected drop of performance between the second model and the first model.

5 Conclusion

We presented a BERT baseline for the Hyperpartisan News Detection task. We demonstrated that pretraining BERT in an unseen domain improves the performance of the model on the domain specific supervised task. We also showed that the difference in news source affects the generalization. Our best performing system ranked 8th out of 42 teams with 78.3% accuracy on the held-out test dataset. From our findings, we believe that domain adaptation is important for the BERT architecture and we would like to investigate the effect of from scratch unsupervised pretraining on the supervised task as future work.

Acknowledgments

We would like to thank task organizers for their support. The study is funded by the European Research Council (ERC) Starting Grant 714868.

Howard Beale

Figure 2: Howard Beale delivering his ”I’m as mad as hell” speech.

Beale[4] is a news anchor who decides to commit suicide on live air. Instead, he gives his famous speech about modern American life and convinces American people to scream his words: ”I’m as mad as hell, and I’m not going to take this any more!”. But the media sees his breakdown as an opportunity for huge ratings. We believe that the speech is now more than ever relevant to our media. Choosing ”Howard Beale” as the team name is our scream from the windows of Academia.

[4]https://www.imdb.com/title/tt0074958/

References

Chris Alberti, Kenton Lee, and Michael Collins. 2019. A bert baseline for the natural questions. *arXiv preprint arXiv:1901.08634*. Version 1.

Marco T Bastos and Dan Mercea. 2017. The brexit botnet and user-generated hyperpartisan news. *Social Science Computer Review*, page 0894439317734157.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. Version 1.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*. Version 1.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*. Version 3.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 231–240. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Elisa Shearer and Jeffrey Gottfried. 2017. News use across social media platforms 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Mark Wasson. 1998. Using leading text for news summaries: Evaluation results and implications for commercial summarization applications. In *COLING 1998 Volume 2: The 17th International Conference on Computational Linguistics*, volume 2.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718*. Version 1.

Chenguang Zhu, Michael Zeng, and Xuedong Huang. 2018. Sdnet: Contextualized attention-based deep network for conversational question answering. *arXiv preprint arXiv:1812.03593*. Version 5.

Team JACK RYDER at SemEval-2019 Task 4: Using BERT Representations for Detecting Hyperpartisan News

Daniel Shaprin[1], **Giovanni Da San Martino**[2], **Alberto Barrón-Cedeño**[2], **Preslav Nakov**[2]
[1]Sofia University "St Klimen Ohridski", Sofia, Bulgaria
[2]Qatar Computing Research Institute, HBKU, Doha, Qatar
shaprin@uni-sofia.bg
{gmartino, albarron, pnakov}@hbku.edu.qa

Abstract

We describe the system submitted by the Jack Ryder team to SemEval-2019 Task 4 on Hyperpartisan News Detection. The task asked participants to predict whether a given article is hyperpartisan, i.e., extreme-left or extreme-right. We propose an approach based on BERT with fine-tuning, which was ranked 7th out 28 teams on the distantly supervised dataset, where all articles from a hyperpartisan/non-hyperpartisan news outlet are considered to be hyperpartisan/non-hyperpartisan. On a manually annotated test dataset, where human annotators double-checked the labels, we were ranked 29th out of 42 teams.

1 Introduction

SemEval-2019 Task 4 (Kiesel et al., 2019) asks to distinguish between articles that are extremely one-sided, i.e., extreme-left or extreme-right, and such that are not. The organizers provided two datasets:

1. **By article:** A small dataset of 645 manually annotated articles (*BA* in the following).
2. **By publisher:** A large dataset of 750,000 articles annotated using distant supervision, where an article is considered hyperpartisan if its source is labeled as such (*BP* in the following). The set is separated into 600,000 articles for training (*BP-train*) and 150,000 articles for validation (*BP-val*).

Furthermore, two test sets, one annotated by article (*BA-test*) and one annotated by publisher (*BP-test*), were hidden from the participants and they were used for getting the final scores for the competition. The task is a binary classification one, where each article is to be assigned one of two possible classes: *hyperpartisan* and *non-hyperpartisan*.

2 Related Work

Media bias was used as a feature for "fake news" detection (Horne et al., 2018a). It has also been the target of classification, e.g., Horne et al. (2018b) predicted whether an article is biased (*political* or *bias*) vs. unbiased. Similarly, Potthast et al. (2018) classified the bias in a target article as (*i*) left vs. right vs. mainstream, or as (*ii*) hyper-partisan vs. mainstream. Left-vs-right bias classification at the article level was also explored by Kulkarni et al. (2018), who modeled both text and URL structure. Some work targeted bias at the phrase or the sentence level (Iyyer et al., 2014), for political speeches (Sim et al., 2013) or legislative documents (Gerrish and Blei, 2011), or targeting users in Twitter (Preoţiuc-Pietro et al., 2017). More recent work has targeted the political bias of entire news outlets (Baly et al., 2018, 2019). Another line of related work focused on propaganda, which is a form of extreme bias (Rashkin et al., 2017; Barrón-Cedeño et al., 2019a,b). See also a recent position paper (Pitoura et al., 2018) and an overview paper on bias on the Web (Baeza-Yates, 2018). Overall, most of the above work focused on finding effective representations, e.g., in terms of features, rather than investigating the impact of sophisticated learning algorithms.

Recently, BERT, a pre-trained deep neural network (Devlin et al., 2019), based on the Transformer (Vaswani et al., 2017), has improved the state of the art for many natural language processing tasks. For example, it reached a score of 80.4 on the GLUE benchmark[1], 86.7% accuracy on MultiNLI, and F_1=93.2 on the SQuAD v1.1 question answering task. Currently, the top 11 systems in the SQuAD v2.0 use BERT.[2]

[1]https://gluebenchmark.com/.
[2]http://rajpurkar.github.io/SQuAD-explorer/.

3 Method

We hypothesize that hyperpartisanship and extreme bias detection are related to sentiment analysis, which is one of the tasks in the GLUE benchmark. Given the recent success of BERT (Devlin et al., 2019) for sentiment analysis and other language processing tasks, we decided to experiment with it for hyperpartisan news detection.

In order to have a reference, we also experimented with Random Forests over TF.IDF representations. We used two BERT models: BERT without fine-tuning, and BERT with fine-tuning. We describe them in more detail below. In each case, we extracted features from the title and from the main text of the articles separately.

3.1 TF.IDF Features

In order to have a reference to compare our BERT-based approaches to, we also experimented with word-level TF.IDF features. First, we converted all text to lowercase and we stemmed it with the Porter stemmer. Then, we removed words with document frequency higher than 0.8. We then extracted two feature vectors by computing the term frequency and the inverse document frequency once on the title and separately on the body of the articles. We ended up with feature vectors of size 110,229 for the title and 1,798,179 for the content when TF.IDF vectors were computed on *BP-train* and *BA*, and 95,806 for the title and 1,507,789 for the content, when *BA* only was used.

We used the feature vectors in a Random Forest classifier with 100 estimators. Note that, differently from BERT, the TF.IDF representation is able to use information from the entire article.

3.2 Pre-trained BERT Features

Our second approach uses features extracted from Google's BERT, a model with pre-training language representations (Devlin et al., 2019). We fed to the model (*i*) the entire title and (*ii*) the first 256 tokens from the body of the article as two separate inputs, and then we obtained vector representations from the last layer of the BERT neural network. Note that we used the pre-trained BERT rather than training it with the data from the competition. Next, we concatenated the vectorial representations and we fed them to a two-layer feed-forward neural network with 32 neurons in the hidden layer. We used $\tanh$ as the activation function and a Gaussian noise with $\sigma = 0.2$.

3.3 Fine-tuned BERT Features

A natural extension of the approach in Section 3.2 is to fine-tune the BERT model on the datasets of the competition, i.e., on *BP+BA*. We performed fine-tuning on the same input used for computing the TF.IDF representations and we obtained two models, one from the titles only and one from the content of the articles only. As we did for the pre-trained model, we concatenated the internal vector representations from the last layer for both models and we passed them to the second neural network as in Section 3.2.

4 Experiments

We performed a number of experiments in order to select the best models to submit as official runs for the competition. The best model for the by-publisher dataset was selected on *BP-val* after training the models on *BP-train*. Since there was no validation set for the by-article dataset and *BA* was too small to be divided into training and validation sets, we trained our models on *BP* and we selected the best-performing one on *BA*. Table 1 shows the obtained accuracy values on *BP-val* (By-publisher) and *BA* (By-article) datasets. As we can observe, the performance of the TF.IDF model is behind those when using BERT, both with and without fine-tuning.

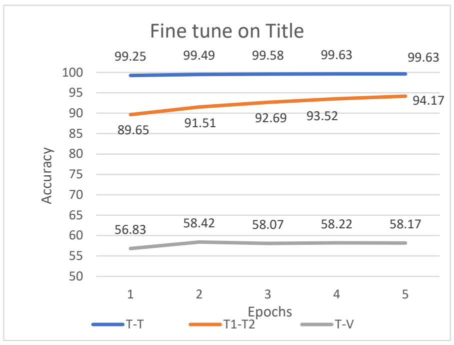
As a result, we opted for the two BERT models, trained on *BP+BA*, as our submissions for the competition. Table 2 shows the results on the hidden test sets.

Model	By publisher	By article
TF.IDF	56.13%	56.12%
BERT (no tuning)	61.20%	60.93%
BERT (fine-tuned)	61.70%	61.30%

Table 1: **Validation results:** Accuracy for our TF.IDF and BERT models on the by-publisher validation (*BP-val*) and on the by-article (*BA*) sets. The training was performed on *BP-train* and *BP*, respectively.

Model	By publisher	By article
BERT (no tuning)	63.25%	64.49%
BERT (fine-tuned)	64.60%	64.50%

Table 2: **Testing results:** Accuracy on the hidden test sets for the BERT models we submitted. Both models were trained on *BP+BA*.



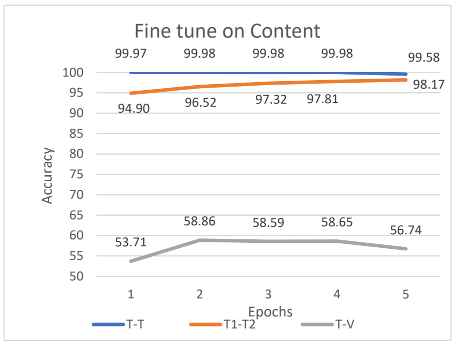


Figure 1: **Title as input:** Accuracy, at each epoch, for the fine-tuned BERT model. T-T stands for training and evaluating on *BP-train*, T1, T2 for training on the first half of *BP-train* and evaluating on the second half, T-V for training on *BP-train* and evaluating on *BP-val*.

Figure 2: **Content as input:** Accuracy, at each epoch, for the fine-tuned BERT model. T-T stands for training and evaluating on *BP-train*, T1, T2 for training on the first half of *BP-train* and evaluating on the second half, T-V for training on *BP-train* and evaluating on *BP-val*.

4.1 Result Analysis and Post-Submission Experiments

When developing the model for the submission, we focused on the datasets with by-publisher annotation. This is probably the reason why we performed much better on the by-publisher hidden test set, 7th out 28 teams, than on the hidden by-article test set, 29th out of 42 teams.

Another possible reason for the low results on the by-article hidden test set is overfitting on *BP*: our model might have learned to discriminate the publishers appearing in *BP* instead of the required labels *hyperpartisan* / *non-hyperpartisan*. Recalling that *BP-val* does not contain any articles from the publishers in *BP-train*, we conducted an experiment to see whether there was a correlation between the articles in the different partitions of the provided dataset. In particular, we created a recurrent model with a single layer of 1,024 GRUs, and we trained it on 80% of the data and we evaluated it on the remaining 20%. The model achieved 99.99% accuracy at predicting whether the article was from *BP-train* or from *BP-val*.

We further performed three additional experiments with the fine-tuned BERT model: (*i*) training and evaluating on *BP-train*, (*ii*) training on the first half of *BP-train* and evaluating on the second half of *BP-train*, and (*iii*) training on *BP-train* and evaluating on *BP-val*.

Figure 1 shows the accuracy for BERT at each epoch when using titles for the three configurations (*i*)–(*iii*) above: T is *BP-train*, T1 and T2 are the two halves of *BP-train*, and V is *BA*.

Figure 2 reports the performance for the same experiments when using the body text of the articles as input. While the accuracy for the curves T-T and T1-T2 is close to 100% or is monotonically increasing with respect to the number of training epochs, the curve T-V does not show the same behavior, suggesting that 58.42% is close to the best performance that can be achieved in this setting. The accuracy values, although not directly comparable, show that there is a huge gap between the performance on a dataset with the same set of publishers (T-T and T1-T2) vs. on a dataset where the news comes from a different set of publishers (T-V), thus supporting our hypothesis.

5 Conclusions and Future Work

We have described our participation in SemEval-2019 task 4 on hyperpartisan news detection. In particular, we explored using TF.IDF and BERT-derived representations, and we found the latter to be more informative. Thus, we submitted two BERT models as our official runs to the competition: one with and one without fine-tuning. Interestingly, fine-tuning the model did not yield any sizable improvements. Our analysis suggests that our BERT models might be learning the source of the article, rather than whether it represents a piece of hyperpartisan news.

In future work, we plan to experiment with the big cased BERT model and to combine it with stylistic features, which have been proven successful for the hyperpartisanship detection task.

References

Ricardo Baeza-Yates. 2018. Bias on the web. *Commun. ACM*, 61(6):54–61.

Ramy Baly, Georgi Karadzhov, Dimitar Alexandrov, James Glass, and Preslav Nakov. 2018. Predicting factuality of reporting and bias of news media sources. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '18, pages 3528–3539, Brussels, Belgium.

Ramy Baly, Georgi Karadzhov, Abdelrhman Saleh, James Glass, and Preslav Nakov. 2019. Multi-task ordinal regression for jointly predicting the trustworthiness and the leading political ideology of news media. In *Proceedings of the 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '19, Minneapolis, MN, USA.

Alberto Barrón-Cedeño, Giovanni Da San Martino, Israa Jaradat, and Preslav Nakov. 2019a. Proppy: A system to unmask propaganda in online news. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, AAAI'19, Honolulu, HI, USA.

Alberto Barrón-Cedeño, Giovanni Da San Martino, Israa Jaradat, and Preslav Nakov. 2019b. Proppy: Organizing news coverage on the basis of their propagandistic content. *Information Processing and Management*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL-HLT '19, Minneapolis, MN, USA.

Sean M. Gerrish and David M. Blei. 2011. Predicting legislative roll calls from text. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML '11, pages 489–496, Bellevue, Washington, USA.

Benjamin Horne, Sara Khedr, and Sibel Adali. 2018a. Sampling the news producers: A large news and feature data set for the study of the complex media landscape. In *Proceedings of the Twelfth International Conference on Web and Social Media*, ICWSM '18, pages 518–527, Stanford, CA, USA.

Benjamin D. Horne, William Dron, Sara Khedr, and Sibel Adali. 2018b. Assessing the news landscape: A multi-module toolkit for evaluating the credibility of news. In *Proceedings of the The Web Conference*, WWW '18, pages 235–238, Lyon, France.

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1113–1122, Baltimore, MD, USA.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan news detection. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, SemEval '19, Minneapolis, MN, USA.

Vivek Kulkarni, Junting Ye, Steven Skiena, and William Yang Wang. 2018. Multi-view models for political ideology detection of news articles. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '18, pages 3518–3527, Brussels, Belgium.

Evaggelia Pitoura, Panayiotis Tsaparas, Giorgos Flouris, Irini Fundulaki, Panagiotis Papadakos, Serge Abiteboul, and Gerhard Weikum. 2018. On measuring bias in online information. *SIGMOD Rec.*, 46(4):16–21.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A stylometric inquiry into hyperpartisan and fake news. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, ACL '18, pages 231–240, Melbourne, Australia.

Daniel Preoţiuc-Pietro, Ye Liu, Daniel Hopkins, and Lyle Ungar. 2017. Beyond binary labels: Political ideology prediction of Twitter users. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, ACL '17, pages 729–740, Vancouver, Canada.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, EMNLP '17, pages 2931–2937, Copenhagen, Denmark.

Yanchuan Sim, Brice D. L. Acree, Justin H. Gross, and Noah A. Smith. 2013. Measuring ideological proportions in political speeches. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, EMNLP '13, pages 91–101, Seattle, WA, USA.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Team Kermit-the-frog at SemEval-2019 Task 4: Bias Detection Through Sentiment Analysis and Simple Linguistic Features

Talita Anthonio
University of Groningen
University of the Basque Country
t.r.anthonio@student.rug.nl

Lennart Kloppenburg
lennartkloppenburg@live.nl

Abstract

In this paper we describe our participation in the SemEval 2019 shared task on hyperpartisan news detection. We present the system that we submitted for final evaluation and the three approaches that we used: sentiment, bias-laden words and filtered n-gram features. Our submitted model is a Linear SVM that solely relies on the negative sentiment of a document. We achieved an accuracy of 0.621 and a f1 score of 0.694 in the competition, revealing the predictive power of negative sentiment for this task. There was no major improvement by adding or substituting the features of the other two approaches that we tried.

1 Introduction

With the growing role of social media in politics it becomes ever more important to safeguard the integrity of information people consume. News articles about important events in the world can affect political choices. It is therefore crucial to pinpoint what information people know to be trustworthy, factual and unbiased. One way to do this is by using a computational system that detects an author's or publisher's bias in a news article. In Potthast et al. (2018) we have seen that it is possible to build such a system by relying on the writing characteristics of a text.

To shed more light on potential linguistic computational methods for hyperpartisan news detection, we present our participation in the SemEval 2019 shared task on hyperpartisan news detection, of which the purpose is to identify whether a news article contains *hyperpartisan* (Kiesel et al., 2019) content. For our contribution, we set out to experiment with various types and levels of features, such as *a*) **sentiment** that could indicate an author's bias (Recasens et al., 2013), *b*) **bias-laden words** such as assertives, factives and hedges, and *c*) part-of-speech (from now on POS) **filtered n–grams**. In the end, we decided to submit a model that only uses the negative sentiment of an article as a feature. We obtained an accuracy of 0.621 and a f1 score of 0.694 on the by-article test set, which resulted to the 30th place in the competition. On the by-publisher test set, the systems accuracy was 0.589 and its f1 score 0.623 (20th place).

2 Related Work

One of the first studies on detecting linguistic bias in online texts were mainly focused on detecting biased language in Wikipedia articles. Despite the domain difference, this task is related to ours because Wikipedia is also a source of information which should contain unbiased language. Systems that were employed for this task used a combination of linguistic features, such as POS n-grams and binary features representing the usage of bias words, assertive verbs, factive verbs, hedges and sentiment features (Recasens et al., 2013; Hube and Fetahu, 2018). Most of these features were derived from existing lexicons. For sentiment features, both studies used a sentiment polarity lexicon from Liu et al. (2005).

A similar set of features was used in Hutto et al. (2015) to detect sentence based bias in news articles. Yet, they obtained sentiment features using the VADER sentiment analysis tool (Hutto and Gilbert, 2014). Because of their focus on sentence-based bias detection and the high relatedness of their study, it was interesting to investigate whether we could use the same features on document-level classification.

The studies that we discussed so far proved that it is possible to detect bias by using simple computationally derived linguistic features. On the other hand, we work with a much larger amount of documents which also come from a different genre.

Because of these aspects, it was fruitful to investigate how effective these features would be.

3 Data

We worked with the data provided by the organizers of the task. We show an overview of the data we used for the competition in Table 1. The sets named 'by-publisher' are automatically labeled using the publisher of the article, whereas the articles from the 'by-article' set were manually labeled through crowd-sourcing (Vincent and Mestre, 2019).

For the competition, we trained our models on the by-publisher training set. We used this data because of its size and the equal frequency distribution of the labels. We took the model with the highest accuracy on the by-publisher validation set for our final submission. After the evaluation period, we submitted several models that were trained on the by-article training data to find out whether we should have submitted a model that was trained on the by-article data.

name	**function**	**size**	**distribution**
by-publisher	training set	600,000	50-50
by-publisher	validation set	150,000	50-50
by-publisher	test set	4,000	50-50
by-article	training set	645	37% hyper. 63% mainstr.
by-article	test set for competition	638	50-50

Table 1: An overview of the data that we used for the shared task.

4 Final System

4.1 VADER Sentiment Analysis

The system we submitted for final evaluation is a simple SVM classifier with a linear kernel. It uses the LinearSVM[1] implementation from *scikit-learn* (Pedregosa et al., 2011) with default hyperparameter settings *C=1.0*, the *'squared hinge'* loss function and the *'l2'* penalization norm. The only features that the classifier uses to make a prediction is the intensity of the negative sentiment of each document which varies between 0 (neutral) and 1.0 (extremely negative). This score is computed by the freely available package **V**alence **A**ware **D**ictionary and s**E**ntiment **R**easoner[2] from NLTK(Loper and Bird, 2002). VADER was developed by Hutto and Gilbert (2014) and is a rule-based and lexical model for general sentiment analysis. Even though VADER is specifically developed to perform sentiment analysis in social media texts, the tool works reasonably well on determining the sentiment of news articles according to our observations.

4.2 Results

The system we submitted reached the 30th place with an accuracy of 0.621. The other scores are reported in Table 2. It also displays the performance of our model on the *by-publisher* development set and the *by-article* test set. We obtained a high recall on the official test set, which corresponds to the performance on the *by-article* test set. Nonetheless, since the evaluation script only tracked the *hyperpartisan=true* class and our model was apparently biased towards a hyperpartisan prediction, this metric is not informative because it implies that the mirrored *hyperpartisan=false* class has a much lower recall.

Furthermore, the scores in Table 2 show that our model neither performed well on the *by-article* (0.519 accuracy) nor the *by-publisher* (0.562 accuracy) development set. In particular, the accuracy on the *by-article* set is even lower than the accuracy of the baseline. Nonetheless, the model performed better on the official test set, since the accuracy and f1 score were substantially higher. We surmise that this is related to the inconsistent similarities of the data sets rather than the predictive power of the features.

	Accuracy	**Precision**	**Recall**	**F1**
official test set	0.621	0.582	0.860	0.694
by-publisher	0.562	0.559	0.585	0.572

Table 2: Evaluation metrics (of the true class) across different data sets of correctly detecting the hyperpartisan class.

5 Alternative Methods

Despite the low performance on the *by-publisher* development set, we submitted our final system because it had the highest accuracy on this development set (0.562 accuracy) and the competition evaluation was based on accuracy. In this section,

[1]https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

[2]https://www.nltk.org/_modules/nltk/sentiment/vader.html

Features	Set-up	Accuracy	Precision	Recall	F1 score
Tf-idf	Word uni-grams with default settings	0.5598	0.5412	0.7854	0.6408
	Word uni-grams with default settings + negative sentiment	0.5597	0.5411	0.7858	0.6409
Sentiment	positive score + negative score + compound	0.5247	0.5201	0.6373	0.5729
	compound score	0.4310	0.4389	0.4962	0.4658
	negative sentiment	0.5616	0.5589	0.5851	0.5717
	positive sentiment	0.4752	0.4779	0.5354	0.5050

Table 3: Performance of other models *trained* on the by-publisher training set on predicting hyperpartisan.

we outline the two other approaches with which we experimented: bias-laden words and filtered n-grams. We also present our attempts to improve the accuracy of our best model on the development set by combining features from the other two approaches. The performance of these models and other detrimental models on the development set are shown in Table 3.

5.1 Sentiment

VADER Sentiment In addition to the negative sentiment score, we also conducted experiments with systems that used several combinations of the negative, positive and compound score provided by VADER. However, none of the combinations outperformed the accuracy of the system that only took the negative sentiment into account. Moreover, as shown in Table 2, the compound score yielded the lowest performance. In our preliminary experiments, we also experimented with the neutral sentiment score but this led to low accuracy scores compared to the other scores. Besides, the evaluation procedure was based on predicting the hyperpartisan is true class, for which we can assume that its corresponding article is not neutral.
We tried to improve the score of the submitted model by using ordinal scales instead of interval variables, in which documents with a negative sentiment score exceeding 0.5 were labeled as having a ”high” negative sentiment and ”low” otherwise. This did not lead to improvement, which reveals that the raw sentiment score is a better predictor.

Other Sentiment Features We also developed systems that calculated the overall sentiment of a text by using the lexicons of positive and negative words from Liu et al. (2005). We experimented with two methods (1) by counting the amount of positive/negative words and (2) by using binary features where the value was True if it contained one of the words in the lexicons. This method was also used in Recasens et al. (2013). Nonetheless both methods did not even reach an accuracy of 30 percent.

5.2 Bias-Laden Words

Verbs We experimented with the same set of verbs as the mentioned previous studies: assertive verbs, factive verbs and hedges (Hooper, 1975). These words carry cues that may indicate bias. For instance, assertive verbs can be used to assert the truth of a proposition (i.e. *point out*, *claim*, *states*) and factive verbs can be used to presuppose the truth of their corresponding complement clause (i.e. *realize*, *revealed*, *indicated*). The usage of these verbs was encoded in the same way as we did for the lexical sentiment features. Yet, it was not possible to build accurate classifiers that used these features, since the accuracy fluctuated between 0.20 and 0.30. Also, we could not increase the performance of other systems by adding these features.

N-gram BOW We additionally tried to derive bias-laden words through BOW methods, as we surmised that the hyperpartisan texts contained more bias-laden words than legitimate news articles. Because of the size of the training set, we only experimented with uni-gram features (with tf-idf weighting). With this set-up, we obtained a similar accuracy score as when we used only the negative sentiment (see Table 2). Yet, we did not submit the uni-gram model for the competition because we surmised that the effectiveness of bag-of-word features would be more sensitive to the topic of the articles. As an effect, the generalizability on unseen data could be low.

5.3 POS-based N-gram Filtering

We experimented with POS-based features early on in an attempt to model *how* and *where* humans would perceive bias in a text on word-level. We found that adjectives, adverbs and (proper) nouns all somewhat contributed to the tone of a text. However, confidently identifying bias proved rather challenging in many cases. Nouns frequently provide thematic and topical information

Training data	System	Accuracy	Precision	Recall	F1 score
by-publisher	Submitted system: negative sentiment only	0.621	0.582	0.860	0.694
	Word uni-grams + negative sentiment	0.605	0.564	0.920	0.700
	POS filtering**	0.657	0.636	0.738	0.683
	Positive score + negative score + compound score	0.611	0.590	0.732	0.653
by-article	Character 3-to-5 grams	0.772	0.825	0.691	0.752
	Word uni-grams	0.755	0.803	0.675	0.734
	POS filtering	0.537	0.522	0.863	0.650

Table 4: Performance of models on the by-article test set submitted after the evaluation period. **only trained on 100k randomly obtained documents of the by-publisher set (with a balanced frequency distribution of labels).

about a text and adverbs and adjectives can indicate a level of subjectivity. Modals such as *would*, *could* and *must* could additionally carry assertiveness that could be related to bias (Recasens et al., 2013; Hube and Fetahu, 2018). We tried modelling this by extracting *n-grams* that followed certain patterns such as *a*) **nouns** in the middle of a trigram *b*) **particles** in the middle of a trigram *c*) **modal** verbs in the middle of a trigram *d*) **nouns** and their closest preceding adjectives/adverbs *e*) **adjectives/adverbs** and words after them. The n-grams essentially became a new, filtered representation of the *document* and would be weighted using tf-idf. We tested this intuition by splitting the training data. The results were quite promising as we (unofficially) achieved f1 scores of 75-85% using only 200,000 documents. However, results disappointingly floated between 53% and 58% when tested on the development data.

We concluded that the *n-grams* we extracted were not indicative enough to generalize well across different data sets, since they were essentially only a subset of the total body of possible *n-grams*.

6 Other Submissions

After the final submission deadline, we continued submitting models to investigate differences between the by-article and by-publisher training data sets. We also submitted models that solely relied on bag-of-words features, with which we experimented in early stages but discarded in our final submission because of the low performances on the by-publisher validation set.

The results of our submissions after the deadline (Table 4) reveal that bag-of-words and bag-of-characters are indeed useful when the model is trained on the by-article data. In particular, we could have obtained a high accuracy in the competition with a model trained on the by-article set, for instance by using character 3-to-5 grams (0.772 accuracy). Another observation is that the POS filtering model obtained a low accuracy on the test set, even when it was trained on the by-article data. Thus, this seems to indicate that bag-of-words are more effective than fine-grained POS filtering.

7 Conclusions

Detecting biased language is a difficult task because of the subjectivity of the task and the subtlety of linguistic context cues. *Bias* is a broad term which can be applied to many different areas and is not solely restricted to politics or economics. Per our own observations, it was difficult to exactly pinpoint the bias of a biased article.

We achieved promising results after our final submission with bag-of-words and bag-of-character n-grams. This indicates that a bag-of-words approach is able to identify token-based patterns in corpora that are related to bias. However, the reliability of a bag-of-words approach does depend on the lexical similarity between training and test data. We demonstrated this through our contradicting results on the provided validation and official test data.

Sentiment proved to be quite a strong feature that can already separate biased from unbiased articles, although more heuristics are needed. This could be combined with the title of the article which, much like sentiment, tells us something about the *entire* article. It could also be interesting to experiment with more general cues about *entire* texts rather than treating texts as only bags-of-words. This could help develop a system that scales better across different corpora and domains.

References

Joan B. Hooper. 1975. On assertive predicates. In *Syntax and Semantics Volume 4*, volume 4, pages 91 – 124. Academic Press, New York.

Christoph Hube and Besnik Fetahu. 2018. Detecting biased statements in wikipedia. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pages 1779–1786, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

C.J. Hutto, Scott Appling, and Dennis Folds. 2015. Computationally detecting and quantifying the degree of bias in sentence-level text of news stories. *HUSO 2015: The first international conference on HUman and Social Analytics*, pages 30–34.

Clayton J. Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *ICWSM*. The AAAI Press.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion observer: analyzing and comparing opinions on the web. page 342–351.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 231–240. Association for Computational Linguistics.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *ACL (1)*, pages 1650–1659. The Association for Computer Linguistics.

Emmanuel Vincent and Maria Mestre. 2019. Crowdsourced measure of news articles bias: Assessing contributors' reliability. In *CEUR Workshop Proceedings*, volume 2276.

Team Kit Kittredge at SemEval-2019 Task 4: LSTM Voting System

Rebekah Cramerus
Department of Linguistics
University of Potsdam
Potsdam, Germany
rebekah.cramerus@fulbrightmail.org

Tatjana Scheffler
Department of Linguistics
University of Potsdam
Potsdam, Germany
tatjana.scheffler@uni-potsdam.de

Abstract

This paper describes the approach of team Kit Kittredge to SemEval 2019 Task 4: Hyperpartisan News Detection. The goal was binary classification of news articles into the categories of "biased" or "unbiased". We had two software submissions: one a simple bag-of-words model, and the second an LSTM (Long Short Term Memory) neural network, which was trained on a subset of the original dataset selected by a voting system of other LSTMs. This method did not prove much more successful than the baseline, however, due to the models' tendency to learn publisher-specific traits instead of general bias.

1 Introduction

With the proliferation of online news agencies after the rise of the Internet, access to information about what is going on in the world has never been more widespread. How that information is presented, however, can have a large influence on what conclusions the reader draws from it. Being able to automatically identify hyperpartisanship (bias or adherence to one party or faction over others) in a news article would help individuals in their news consumption and potentially result in a better-informed population.

As suggested, the challenge approached in this paper is that of hyperpartisan news detection: a binary classification problem (biased or unbiased) with news articles as data. This task can be considered as related to stance detection and in general, sentiment analysis. The challenge was organized as Task 4 for SemEval 2019 (Potthast et al., 2018) (Kiesel et al., 2019). Final submissions were submitted through TIRA, with the test datasets hidden (Potthast et al., 2019).

First in this paper, Section 2 includes an introduction of the provided dataset and a description of preprocessing techniques used for our approach. Section 3 describes the first submitted software, a bag-of-words model. Section 4 continues with our second approach, an LSTM trained on a subset of the original dataset, and a description of how that subset was selected.

Section 5 presents our results on the test set and Section 6 delves into analysis, presenting potential reasons why the models did not perform very well.

2 Data

During the course of the task participants were granted access to different datasets with which to work.

A training dataset of 600,000 articles and a validation dataset of 150,000 articles were both released to participants. Both sets contain 50% unbiased and 50% biased articles, and of the latter, half are left-biased and the other half right-biased (in terms of their placement on the political spectrum). Importantly, these articles were all labeled with the overall bias of their publisher, which was obtained by MediaBiasFactCheck.com and BuzzFeed. The set of publishers whose articles appear in the training set has no overlap with the publishers of the validation set, and neither has any overlap with the publishers whose articles appear in the inaccessible test set.

We consider the labels of these datasets to be noisy: though publishers may have an overall bias, it is likely that most biased news agencies do not publish only biased articles, just as most unbiased news agencies may occasionally publish a biased piece.

Also relevant is a third released dataset referred to in this paper as the *byarticle* dataset. Unlike the other datasets, this one contains articles which were labeled individually through crowdsourcing. It is small, at 645 articles, and unbalanced, at 63% unbiased and 37% hyperpartisan.

2.1 Preprocessing

Certain preprocessing tasks were carried out on the entire dataset pre-training, and also applied to the test set during evaluation.

Some cleaning tasks required segmentation of texts into sentences or words using the Natural Language Toolkit (NLTK) (Bird et al., 2009).

Special characters, double spaces, more than three dots in a row, and any failures in character translation (for example, "gun control" becoming ?gun control?) were replaced or removed. Regular expressions were used for some of these tasks, as well as for removing `img` or `html` tags and URLs. Another list of phrases were summarily removed from each text: those which were likely byproducts of the articles' retrieval from their websites. These included "Continue Reading Below...", "Image Source:", "Opens a New Window", and so on.

As will be discussed, a recurring problem encountered by our models was the tendency to learn publisher-specific traits and not hyperpartisanship itself. To combat this we included methods to remove potential publisher-specific text, especially names and emails of the authors of the articles, in our preprocessing step.

3 Software 1: Bag-Of-Words

Our first approach was a bag-of-words baseline for initial exploration of the dataset and comparison with the more complex second approach.

3.1 Tokenization / Lemmatization

Texts were tokenized into words, and the words reduced to their lemmas, using spaCy (Honnibal and Johnson, 2015).

3.2 Vectorization

First we created a vocabulary of the most common 4000 words in the overall corpus (all datasets), excluding stopwords from a list compiled by scikit-learn (Pedregosa et al., 2011).

We also excluded from the vocabulary words from an exceptions list, in an attempt to reduce the problem of overfitting to the article publishers. This exceptions list was formed by counting all words in the training and validation datasets, and gathering those words which appeared five times more often (relative to the size of the corpus) in one set than in the other. Some of these terms were location-specific (*abq*, *lobos*, *nmsu* — likely a publisher based in New Mexico) and others hinted at coverage of a certain topic (*samsung*, *boeing*, *verizon* — possibly a publisher which wrote often about the stock market). The intent behind this was to help avoid the model picking up, for example, that the presence of terms surrounding New Mexico automatically meant a certain label.

With a final vocabulary, each text was then converted to a vector of length 4000, wherein each dimension is the count of the corresponding vocabulary word in that text.

3.3 Model

Using scikit-learn, the training and validation datasets were vectorized according to the previously described specifications and then fit to a logistic regression model. This was then submitted to TIRA.

4 Software 2: LSTM

Long Short Term Memory networks (or LSTMs) are a form of Recurrent Neural Networks (or RNNs) which is capable of learning long-term dependencies. Given the nature of the problem and the data — a text being a sequence of words, the relationship between them as important as the words themselves — we chose to develop a model with this architecture.

4.1 Word Embeddings

To transform the article texts into vectors able to be processed by the neural network, we chose to train our own skip-gram word embeddings on the total given corpus (training, validation and byarticle datasets). Embeddings of 50 dimensions (chosen mostly arbitrarily, but in part due to computational limits) were trained using the Python package gensim, for 10 epochs, including words in the vocabulary which appeared in the corpus over five times (Řehůřek and Sojka, 2010). The total vocabulary size was 457,197 words.

4.2 Vectorization

Texts were first transformed into arrays of the shape (100, 50), wherein 100 was the cutoff or maximum text length and 50 was the dimensionality of the word embeddings. Texts shorter than 100 words were padded with zero-vectors to keep the shape consistent to feed into the network.

Model	Test Dataset	Accuracy	Precision	Recall	F1
Bag-of-Words	byarticle	57.8	54.7	90.8	68.3
LSTM	byarticle	58.3	55.8	79.2	65.5
Bag-of-Words	bypublisher	61.2	57.8	83.4	68.2
LSTM	bypublisher	65.2	64.7	67.1	65.9

Table 1: Results on the test datasets.

4.3 Architecture

The model consists of a single LSTM layer with 50 units, followed by a dropout wrapper with a keep probability of 0.75 to help prevent overfitting. Next is a standard feedforward neural network output layer. AdamOptimizer was used with a 0.001 learning rate as well as softmax cross-entropy loss for optimization.

All LSTM models were trained using Tensorflow for approximately 2 epochs (Abadi et al., 2015).

4.4 Voting System

Knowing that the biggest obstacle faced so far was the tendency of models trained on the datasets to overfit to the publishers and not bias itself, we chose to pare down the dataset for the final submission. In theory the ideal dataset, in which there is no noise from the publisher-based labels, is contained within the original dataset. To find that subset — or at least to get closer — we implemented a voting system.

Three LSTMs of the previously described architecture were trained: one each on the training, validation and byarticle datasets. We then collected predictions from each LSTM, on each article in each dataset. The articles which all three LSTM models correctly labeled were pulled into a new dataset labeled *agree*. This dataset, in total size 162,046 articles with 37% biased and 63% unbiased labels, was what we trained our final model on.

4.5 Retrained LSTM

Once the new datasets were compiled from the voting system based on the originals, a new LSTM with the same architecture was trained on the combined data. This model was submitted to TIRA as our second software.

5 Results

Both approaches were scored on the hidden test datasets using TIRA. One of the two test datasets was labeled individually by article, referred to in this paper as the *byarticle-test* dataset, including 638 articles with no publisher overlap with any of the given corpora. The other, like the training and validation sets, was labeled overall by publisher, here referred to as the *bypublisher-test* dataset, with a total of 4000 articles, also including no publisher overlap with other datasets.

Results can be seen in Table 1. The LSTM tended to outperform the Bag-of-Words model in accuracy, but had lower f1 scores. All results showed higher recall than precision — and except for the case of the LSTM with the bypublisher-test set, markedly higher. By accuracy, the best result was the LSTM on the bypublisher-test set.

6 Discussion

In the published leaderboard, most teams had higher scores on the test dataset which was labeled by article rather than the one labeled by publisher; overall, the highest accuracy was over 10% higher on the byarticle-test set than on the bypublisher-test one. Our approaches, on the other hand, both performed better on the bypublisher-test dataset. This could be in part because we did not spend too much time optimizing over the small byarticle dataset which was released to us — trying additional techniques to maximize performance over this set could be a task for future work.

Why our results are better on the bypublisher-test sets is an interesting question. Our efforts in both approaches were focused on enabling the models into generalizing about bias, instead of on recognizing only which articles belong to which publishers. Better performance on the bypublisher-test run than on the byarticle-test run suggests that our efforts may have paid off, but in the sense that we are better able to identify biased publishers instead of biased articles. That is, the question that the first models were answering was, "Does this article belong to X set of publishers, or Y set of publishers?" We attempted to instead answer the question, "Is this article biased?" But higher accuracy on the bypublisher-test dataset indicates that we might instead answer the question, "Does this article belong to a biased *publisher*?"

6.1 Precision/Recall

Across all models recall was consistently higher than precision. Our approaches therefore were correctly picking out hyperpartisan articles, but also misclassifying unbiased articles as biased. There are a variety of reasons why this could happen: quotes by partisan speakers could affect a rating of an unbiased article which discusses it, or certain topics could be more often reported in a partisan manner so that unbiased articles around them are rare and misclassified. A closer examination of the test dataset results would be needed for a more concrete discussion.

6.2 Software 1 Exceptions List

The exceptions list was created with the intent of removing words from the vocabulary which were extremely lopsided in their use between publishers (as in, some publishers, or just one in particular, were much more likely to use the term than others). While initial results looked promising, it is possible that the method was not robust enough, and some unigram indicators of a certain publisher still ended up in the final model.

6.3 Software 2 Dataset: Voting System

The idea behind the voting system was to pare down the original dataset, reducing noise and therefore focusing on the data points where bias was most salient — and could as such be picked up by models trained on different publishers. Theoretically these articles would all have characteristics common to biased articles of all publishers. When put together, then, the hope was that a model trained on this subset of the dataset would learn those common characteristics and not just the publisher-specific ones.

There are many things that could have gone wrong with this system, however. Our cutoff for the news article length may have been too short, for example. Secondly, Since the three models used for voting did not have very high accuracy on each other's datasets in the first place, the level of noise may not have been reduced at all. Furthermore, the original training dataset was four times as large as the validation dataset, and the byarticle dataset was far smaller than either. Their subsets after the voting system was applied were equally unbalanced. When combined and used for training the final LSTM, it could have been unbalanced enough that the model learned mostly from the data from the original dataset, and the features from those publishers.

7 Conclusion

In approaching Task 4 (Hyperpartisan News Detection) in SemEval 2019, we developed two models for submission: a Bag-of-Words logistic regression model and an LSTM neural network trained on a subset of the original training and validation sets. While neither model reached high accuracy rates on the test datasets, their methods still provoke some discussion on how to better avoid fitting to the publishers and not bias itself.

8 Namesake

Margaret Mildred "Kit" Kittredge is a character from the American Girl doll and book series. She was born in Ohio in the 1920s and wanted to become a reporter when she grew up. In her room in the attic, she would write her news articles on a typewriter to share with her family.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Steven Bird, Edward Loper, and Ewan Klein. 2009. Natural language processing with python. O'Reilly Media Inc.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal. Association for Computational Linguistics.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. TIRA Integrated Research Architecture. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 231–240. Association for Computational Linguistics.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

Team Ned Leeds at SemEval-2019 Task 4: Exploring Language Indicators of Hyperpartisan Reporting

Bozhidar Stevanoski, Sonja Gievska
Faculty of Computer Science and Engineering
Ss. Cyril and Methodius University
Rugjer Boshkovikj 16, Skopje, Republic of North Macedonia
bozidar.stevanoski@students.finki.ukim.mk
sonja.gievska@finki.ukim.mk

Abstract

This paper reports an experiment carried out to investigate the relevance of several syntactic, stylistic and pragmatic features on the task of distinguishing between mainstream and partisan news articles. The results of the evaluation of different feature sets and the extent to which various feature categories could affect the performance metrics are discussed and compared. Among different combinations of features and classifiers, Random Forest classifier using vector representations of the headline and the text of the report, with the inclusion of 8 readability scores and few stylistic features yielded best result, ranking our team at the 9^{th} place at the SemEval 2019 Hyperpartisan News Detection challenge.

1 Introduction

Current influential technological megatrends, such as, smart phones and social networking often come with some unwanted side effects - prolific spread of false, biased and misleading information, for instance. In the past few years, the potential threats and consequences of disseminating fake news has reinforced the discussion on the responsibility of social media and governments to tackle the issue, sooner rather than later. Enabling users to report on and be informed of untruthful, deceitful and fraudulent content and sources is expected to become a type of guiding principle for those involved in publishing and disseminating content online.

There is a blurred line between deceptive writing and hyperpartisan reporting, producing extremely biased articles in favor of one political party, cause or individual, while preserving the format and appearance of professional articles. Adherence to the ethics and rules of objective reporting is frequently debatable when it comes to political analysis in media articles. While certain truthful facts are present, they are carefully entangled in a narrative package with biased views, populistic messages and divisive topics, using language that polarizes and flares emotions. Rather than labeling and grading news articles on the truth continuum, researchers usually opt for identification of the phenomenological and contextual features of distinguishing hyperpartisanship in online news articles.

People use diverse set of cues extrapolated from published text and external knowledge and sources, when verifying the veracity of information imparted by others. A large body of evidence documents the impact of deception has on language choices people make. A notable body of work exists revealing insights into the language of deceit in interrogation context (Porter and Yuille, 1996), court hearings (Coulthard et al., 2016), or personal relationships (Miller et al., 1986). Empirical studies still remain the primary manner in which manifestation of deceptive human behavior online is studied. Analysis of political language (Rashkin et al., 2017), partisan media (Gervais, 2014), and news publishing (Rubin et al., 2015) were also guided broadly by the questions pertaining to detecting deception in written language.

In what follows, we highlight the primary findings of our empirical research in identifying tangible verbal indicators as they relate to our central commitment of detecting deception in text. In this paper we examine the impact of grammar and psycho-linguistic word categories, syntactic word connotations and text complexity metrics on the task of distinguishing hyperpartisanship in real news articles.

2 Related Work

Given how prolific fake content has become, the phenomenon has challenged the interdisciplinary research community and has been the focus of notable research studies, especially in the field of natural language processing (NLP) and social network analysis.

While it is beyond the scope of this paper to exhaust a review on the topic, of particular relevance to the authors of this paper are the works in monitoring and detecting what is considered untruthful and deceitful content. The differences in the type of conveyed text and the underlying context are likely to afford contrasting models of deception i.e., combination of linguistic features and selection of classification algorithm they rely upon.

It is interesting to note that rather simple linguistic analysis could be successful on a number of NLP tasks relating to detection of deceptive text, such as fake news, opinions, trolling, hate and abusive language, including hyperpartisan reporting. This indicates that it is not semantics, but rather syntactic and pragmatics of the language style of the author that give clues of the underlying cognitive states relating to deception.

Low-level linguistic features such us word counts and frequencies (Horne and Adali, 2017), language modeling (Conroy et al., 2015; Potthast et al., 2018; Pérez-Rosas et al., 2018), part-of-speech tags (POS) (Lim et al., 2018; Conroy et al., 2015), Probabilistic Context Free Grammar (Feng et al., 2012), readability scores (Potthast et al., 2018), and their combinations have proved to be successful with varying performance and generalization power, especially for testing on cross-domain datasets. The research study most closely related to ours, proposes a model for hyperpartisan classification that yielded accuracy of 0.75 (Potthast et al., 2018), which will be used as a baseline accuracy against which our model will be compared.

The use of deep learning architectures (Wang, 2017) have complemented the list of traditional machine learning algorithms (ML), such as SVM (Yang et al., 2017; Lim et al., 2018), logistic regression, discriminant analysis, decision trees (Potthast et al., 2018) and neural networks (Vuković et al., 2009), used in the field of deceptive detection. An unavoidable discussion on the trade-offs between generality and specificity of the models has never ceased to flavor the interpretation of results and point out directions for future improvements.

3 Dataset

Two datasets of news articles were available for the SemEval 2019 Task 4: ”Hyperpartisan news detection” (Kiesel et al., 2019), one labeled *”by-article”* by professional journalists, and the other labeled *”by-publisher”*.

Our empirical study was focused on the former one, whose training dataset consists of 645 articles. The testing dataset, which is not publicly released, are made available via TIRA (Potthast et al., 2019), and it contains 628 *by-article* articles. It is balanced and consists of articles from previously unseen publishers in the training sets. For evaluation purposes, we randomly choose 80% of the *by-article* data for training, and the remaining 20% for validation.

4 Our Methodology

In this paper, we further enhance the feature set explored by related research, and explore few features that appeared to be promising to capture syntactic and pragmatic aspect of hyperpartisan reporting.

Word vector representations: Though previous research studies on this topic use language modelling i.e., frequencies of n-grams in an article to unmask the style of hyperpartisan reporting, our view is that it is distributed word vector representations might augment the model in capturing the style of deceptive and biased political reporting.

Word2Vec has been emphasized as providing better performance, generalizability and transfer of knowledge on a number of related NLP problems. In consequence, word2vec, pre-trained on part of the Google News dataset consisting of cca 100 billion words (Mikolov et al., 2013) was utilized in our model.

Indication of hyperpartisan language and style could be found in various parts of a journal article - article headline and individual sentences could be indicative of biased and partisan language. Transmission of context, set by a sentence that is entailed in the consecutive sentences in a document, is the core idea underlying the proposed word vector representations on two different levels, one on a sentence level and another on a document/article level. Consequently, three word embeddings representing the headline, the sentences and the entire document text were concatenated creating the final word2vec vector.

For the word and sentence tokenization we use the Natural Language Toolkit (NLTK)[1].

Readability scores: Readability scores measure the ease of comprehension of a particular style of writing based on metrics such as, word and sentence length and various weighting factors and ratios, making them closely related to the quantitative aspect of text complexity. In accordance with successful practices reported in previous research in text deception detection (Pérez-Rosas et al., 2018; Yang et al., 2017), we use eight such scores[2], namely Flesch Reading Ease (Flesch, 1948), Flesch Kincaid Grade Level (Kincaid et al., 1975), Coleman Liau Index (Coleman and Liau, 1975), Gunning Fog Index (Gunning, 1952), SMOG Index (Harry and Laughlin, 1969), ARI Index (Senter and Smith, 1967), LIX Index (Björnsson, 1968) and Dale-Chall Score (Chall and Dale, 1995).

General stylistic measures: We also employ elementary measures - number of characters, total words, different words, sentences, syllables, polysyllable words, difficult words (as defined by (Dale and Chall, 1948)), and words longer than 4, 6, 10 and 13 characters.

Psycho-linguistic features: Motivated by previous studies in the field of deceptive text analysis, including fake news examination (Cunha et al., 2018), exploring fraudulent hotel reviews (Fast et al., 2016), characterizing and detecting hateful Twitter users (Ribeiro et al., 2018), we explore the effect of all 194 types of features from the Empath (Fast et al., 2016) lexicon on the task of hyperpartisan news detection.

Part-of-speech tagging: The frequencies of part-of-speech (POS) categories of the words in text, in particular frequencies of nouns, proper singular nouns, personal and possesive pronouns, wh-pronouns, determiners, wh-determiners, cardinal digits, particles, interjections, adjectives, verbs in base form, past tense, gerund, past participle, 3rd and non-3rd person singular present, were added to our model.

Augmented stylistic feature set: Instead of eliminating stop-words, we take the number of their occurrences as a feature. We use the corpus made available by NLTK. Frequencies of interrogative (how, when, what, why) and all-caps words, negations (not, never, no) and punctuation marks are as stylistic features. The stylistic features were normalized by article length.

Bag-of-words of hyperlinks: The links in each article are abbreviated to their base URL form, using Python's Urllib[3], and further transformed into a bag-of-words (BoW) representation. Both internal (anchor links) and external links in respect to the articles, are taken into account for the BoW representation.

5 Results and Discussion

The relationship between various predictive models and evaluation metrics has always been a topic of interest in machine learning and NLP, and this section describes the performance of the feature sets we have experimented with. It is important to note that since the features we test can take values from different ranges, we perform min-max normalization on all of them to bring them in the $[0, 1]$ interval.

We have experimented with various classifiers, such as Logistic Regression, Multilayer Perceptron and Extra Trees, although the most successful one was Random Forest (RF) classifier with 100 trees, which is in line with the findings of the baseline model (Potthast et al., 2018). We use the Python implementation of the classifiers from the Scikit-learn library. [4]

While aiming for achieving high accuracy, avoiding overfitting was also an objective to ensure the model is robust enough to handle previously unseen data. The evaluation results obtained by the models on *by-article* validation and test datasets are presented in Table 1. A short description of the evaluated models follows:

- **Model 1** - A model that incorporates three concatenated word representation vectors, eight readability scores and the general stylistic features
- **Model 2** - The set of features of Model 1 augmented with psycho-linguistic features
- **Model 3** - Frequencies of the POS tags and additional stylistic features were added to the set of features included in Model 2
- **Model 4** - An extension of Model 1 feature set that included hyperlink features

[1]https://www.nltk.org/ Last accessed: 23 February 2019.

[2]https://pypi.org/project/ReadabilityCalculator. Last Accessed: 20 February 2019

[3]https://docs.python.org/3/library/urllib.html Last Accessed: 23 February 2019.

[4]https://scikit-learn.org/ Last accessed: 23 February 2019.

Models	***By-article* test dataset**				***By-article* validation dataset**			
	accuracy	*precision*	*recall*	*F1 score*	*accuracy*	*precision*	*recall*	*F1 score*
1	0.775	0.865	0.653	0.744	0.837	0.857	0.652	0.741
2	0.769	0.860	0.643	0.736	0.798	0.833	0.543	0.658
3	0.760	0.844	0.637	0.726	0.814	0.844	0.587	0.692
4	0.763	0.851	0.637	0.729	0.837	0.903	0.609	0.727
5	0.710	0.784	0.580	0.667	0.806	0.784	0.630	0.699

Table 1: Performance comparison of models trained on the *by-article* dataset.

- **Model 5** - Principal Component Analysis (PCA) was used to reduce dimensionality of Model 3 to 50 features

The model i.e., feature set that exhibits the best performance is Model 1, that outperforms the other models on the validation as well as on the *by-article* test dataset, but also outperforms the baseline accuracy results presented in (Potthast et al., 2018) by 2.5%.

The attempts to improve the performance on the same dataset by augmenting the set of features with new ones were dissatisfactory and did not lead to any performance advantage. Augmenting the feature set with psycho-linguistic features or POS tags in Model 2 and Model 3 respectively, failed to gain any performance advantage compared to Model 1 Model 4 yielded the worst results. Reducing the dimensionality of the feature space of Model 3 to a 50-dimensional one by using PCA in Model 5, led to even greater degradation of performance metrics. When testing the predicting power of the hyperlink features independently from all other features, the results were significantly better than chance.

The weakness of the models can be explained by the difficulty in defining general heuristics with which to detect biased and deceptive reports. Much of this research represents an effort to understand the clues which give insight into the underlying conditions pertaining to such reporting in news articles. Close inspection of data and comparative analysis with the models participating on the same SemEval task could better support the interpretation of our results. In addition, not having information on the cases that were misclassified by our models, makes it difficult to speculate and offer solutions for proper treatment and improvement of the limitations of our model.

6 Conclusion

In this paper, we report on an experiment that examines the predictive effect of the different feature sets on automatic detection of hyperpartisan articles. Results implicate that the features examined in this research, to varying degree, capture the syntactic and pragmatic aspects of hyperpartisan style, and generalize well to a set of previously unseen articles by unseen publishers. The findings provide evidence of strong modeling capability of word vector embeddings combined with text complexity metrics of the reports and psycholinguistic features, demonstrating that the model accuracy rivals the performance of other teams participating in the SemEval 2019 hyperpartisan challenge, positioning our team at the 9^{th} place on the task's leaderboard.

References

Carl-Hugo Björnsson. 1968. *Läsbarhet: hur skall man som författare nå fram till läsarna?* Bokförlaget Liber.

Jeanne Sternlicht Chall and Edgar Dale. 1995. *Readability revisited: The new Dale-Chall readability formula*. Brookline Books.

Meri Coleman and Ta Lin Liau. 1975. A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60(2):283.

Niall J Conroy, Victoria L Rubin, and Yimin Chen. 2015. Automatic deception detection: Methods for finding fake news. In *Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community*, page 82. American Society for Information Science.

Malcolm Coulthard, Alison Johnson, and David Wright. 2016. *An introduction to forensic linguistics: Language in evidence*. Routledge.

Evandro Cunha, Gabriel Magno, Josemar Caetano, Douglas Teixeira, and Virgilio Almeida. 2018. Fake news as we feel it: perception and conceptualization of the term fake news in the media. In *International Conference on Social Informatics*, pages 151–166. Springer.

Edgar Dale and Jeanne S Chall. 1948. A formula for predicting readability: Instructions. *Educational research bulletin*, pages 37–54.

Ethan Fast, Binbin Chen, and Michael S Bernstein. 2016. Empath: Understanding topic signals in large-scale text. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 4647–4657. ACM.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 171–175. Association for Computational Linguistics.

Rudolph Flesch. 1948. A new readability yardstick. *Journal of applied psychology*, 32(3):221.

Bryan T Gervais. 2014. Following the news? reception of uncivil partisan media and the use of incivility in political expression. *Political Communication*, 31(4):564–583.

Robert Gunning. 1952. *The technique of clear writing*. McGraw-Hill, New York.

McLaughlin G Harry and M Laughlin. 1969. Smog gradinga new readability formula. *Journal of Reading*, 12(8):639–646.

Benjamin D Horne and Sibel Adali. 2017. This just in: fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. In *Eleventh International AAAI Conference on Web and Social Media.*

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. *Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel*. Institute for Simulation and Training, University of Central Florida.

Sora Lim, Adam Jatowt, and Masatoshi Yoshikawa. 2018. Understanding characteristics of biased sentences in news. *INRA 2018, October 2018, Turin, Italy*.

Tomas Mikolov, Kai Chen, G.s Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*, 2013.

Gerald R Miller, Paul A Mongeau, and Carra Sleight. 1986. Invited article fudging with friends and lying to lovers: Deceptive communication in personal relationships. *Journal of Social and Personal Relationships*, 3(4):495–512.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic detection of fake news. *Proceedings of the 27th International Conference on Computational Linguistics, pages 33913401 Santa Fe, New Mexico, USA, August 20-26, 2018.*

Stephen Porter and John C Yuille. 1996. The language of deceit: An investigation of the verbal clues to deception in the interrogation context. *Law and Human Behavior*, 20(4):443–458.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. TIRA Integrated Research Architecture. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 231–240. Association for Computational Linguistics.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2931–2937. Association for Computational Linguistics.

Manoel Horta Ribeiro, Pedro H Calais, Yuri A Santos, Virgílio AF Almeida, and Wagner Meira Jr. 2018. Characterizing and detecting hateful users on twitter. In *Twelfth International AAAI Conference on Web and Social Media.*

Victoria L Rubin, Niall J Conroy, and Yimin Chen. 2015. Towards news verification: Deception detection methods for news discourse. In *Hawaii International Conference on System Sciences.*

RJ Senter and Edgar A Smith. 1967. Automated readability index. Technical report, CINCINNATI UNIV OH.

Marin Vuković, Krešimir Pripužić, and Hrvoje Belani. 2009. An intelligent automatic hoax detection system. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 318–325. Springer.

William Yang Wang. 2017. "liar, liar pants on fire": A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 422–426. Association for Computational Linguistics.

Fan Yang, Arjun Mukherjee, and Eduard Dragut. 2017. Satirical news detection and analysis using attention mechanism and linguistic features. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1979–1989. Association for Computational Linguistics.

Team *Peter Brinkmann* at SemEval-2019 Task 4: Detecting Biased News Articles Using Convolutional Neural Networks

Michael Färber
University of Freiburg, Germany
`michael.faerber@cs.uni-freiburg.de`

Agon Qurdina
University of Prishtina, Kosovo
`agon.qurdina@studentet.uni-pr.edu`

Lule Ahmedi
University of Prishtina, Kosovo
`lule.ahmedi@uni-pr.edu`

Abstract

In this paper, we present an approach for classifying news articles as *biased* (i.e., *hyperpartisan*) or *unbiased*, based on a convolutional neural network. We experiment with various embedding methods (pretrained and trained on the training dataset) and variations of the convolutional neural network architecture and compare the results. When evaluating our best performing approach on the actual test data set of the SemEval 2019 Task 4, we obtained relatively low precision and accuracy values, while gaining the highest recall rate among all 42 participating teams.

1 Introduction

Hyperpartisan news detection describes the task of given a news article text, decide whether it follows a hyperpartisan argumentation, i.e., whether it exhibits blind, prejudiced, or unreasoning allegiance to one party, faction, cause, or person (Kiesel et al., 2019). In recent years, hyperpartisan news detection, which we consider synonymous to news bias detection, has attracted the interest of researchers and various approaches for news bias detection have been developed (Recasens et al., 2013; Baumer et al., 2015; Baly et al., 2018). However, the definition of *bias* and the task set-up of identifying biased news articles differs from authors to authors. For instance, authors might consider the bias in terms of the writing style, while others might consider it in relation to fact selection (Hamborg et al., 2018). In this paper, we use the definition and data set of SemEval 2019 Task 4 (Kiesel et al., 2019), which deliberately uses the generic definition outlined at the beginning. Note that news bias detection differs from related tasks such as opinion finding (Ounis et al., 2006; Macdonald et al., 2007), sentiment analysis, fake news detection (Potthast et al., 2018), claim assessment (Popat et al., 2016), argumentation mining on news articles (Palau and Moens, 2009), and personality detection based on texts (Mairesse et al., 2007).

From a technical perspective, in recent years deep learning techniques have outperformed traditional methods concerning various NLP tasks. This also applies to news article classification tasks (Ounis et al., 2006; Macdonald et al., 2007). Indeed, in the SemEval Twitter sentiment analysis competition in 2015, 2016, and 2017 (Rosenthal et al., 2015; Nakov et al., 2016; Rosenthal et al., 2017), among the most popular (and apparently effective) deep learning techniques were convolutional neural networks (CNNs). Thus, we decided to build a hyperpartisan news classifier based on a CNN. Next to our basic model, we also develop and evaluate variations of our model.

2 Approach

In the following, we outline our approach for news bias detection.[1]

2.1 Preprocessing

Given a set of news articles as input, we preprocessed them along the following steps:

Text Cleaning. We replaced the new lines by spaces, expanded contractions, and removed stop words, HTML tags, and special characters from the articles' content.

Texts to Sequences. The articles' content was tokenized and a word dictionary (size: 1,207,438) was generated.

Sequence Padding. We applied a padding with

[1]Note that our team's name is dedicated to *Peter Brinkmann*, the German journalist who asked the question that ultimately fractured the Berlin wall in 1989. The source code of the implementation is available online at `https://github.com/michaelfaerber/SemEval2019-Task4`.

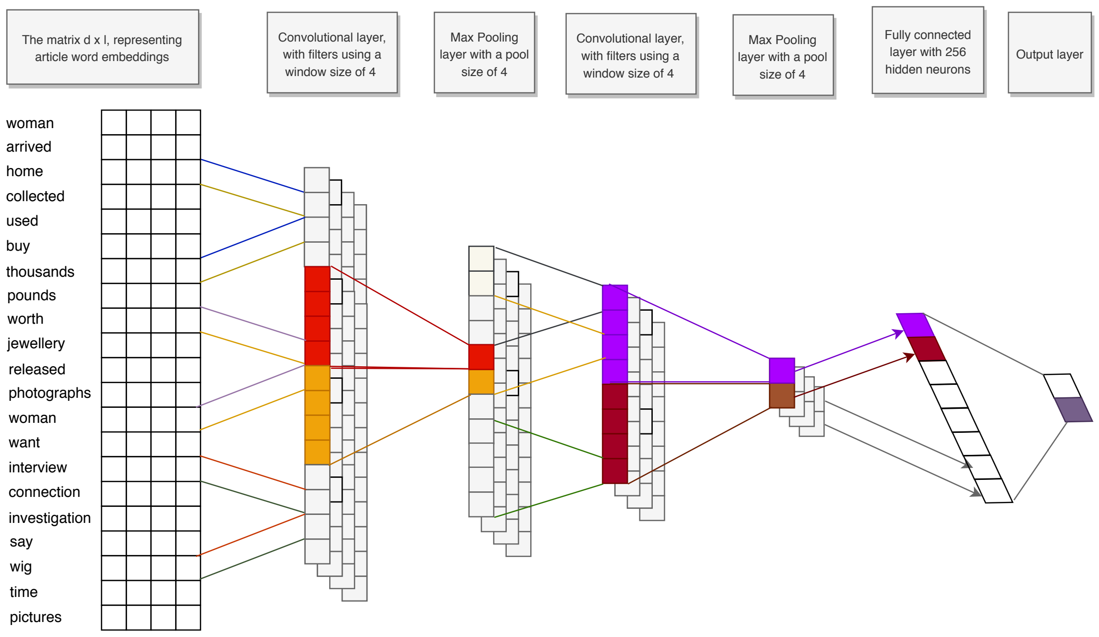

The matrix d x l, representing article word embeddings
Convolutional layer, with filters using a window size of 4
Max Pooling layer with a pool size of 4
Convolutional layer, with filters using a window size of 4
Max Pooling layer with a pool size of 4
Fully connected layer with 256 hidden neurons
Output layer
woman
arrived
home
collected
used
buy
thousands
pounds
worth
jewellery
released
photographs
woman
want
interview
connection
investigation
say
wig
time
pictures

Figure 1: Architecture of our system used for classifying sentences.

a fixed sequence length l. We set $l = 5000$. Thus, around 0.04% of the sequences were truncated.

2.2 Basic Model Architecture

Our basic architecture is shown in the Figure 1. We use a CNN architecture that is based on Kim et al.'s approach for sentence classification (Kim, 2014). His proposed architecture has been widely applied for various tasks in the past. The CNN consists of two subsequent one-dimensional convolutional neural networks layers, appended by MaxPooling layers, and a dense neural networks layer processing the output of the second CNN layer. The model is completed by a final output layer that uses the sigmoid activation function to return a binary output (i.e., the classification into *biased* or *non-biased*).

In the following, we describe the architecture in more detail:

Input layer. Considering article's words were embedded into d-dimensional vectors, the final matrix used as input to the model can be written as $I = l \times d$ where l is the chosen sequence length (i.e., the length of the articles). Recall that $l = 5000$ in our setting.

First convolutional layer. The first transformation this embedded input goes through is a convolutional layer with f 1-dimensional filters of length k. Thus, the layer weights can be considered a matrix of shape $Wc \in R^{f \times k}$.

We chose as filter size $f = 64$, while using a filter length of $k = 4$. In our context, having 1-dimensional filters means that for each word in an article, its three adjacent words are considered as the context of the word. The output of the convolutional layer then is $C = conv(I, Wc)$ where $conv$ is the convolutional operation applied to input I using the weights matrix Wc. This operation includes applying the *ReLu* activation function to complete weights calculations. Also, a dropout function is used to prevent overfitting. We used a dropout rate of 0.2, which means 20% of the weights, chosen randomly, during each training epoch are set to 0.

First max pooling layer. The above output C is then considered the input to a 1-dimensional Max Pooling layer. The purpose of the layer is to try extracting only the most important features of the convolution outputs. This is done by keeping only the max value from a pool size p. As we chose $p = 4$, the output of this layer can be written as $M = max\_pool(C, p)$. This operation reduces the number of weights by four times.

Second convolutional layer and max pooling layer. The output M of the max pooling layer is the input of the second convolutional layer, and the whole process described above is applied to this input, to get a final output M_2.

Fully connected layer. Given the two-dimensional matrix of weights from the last step, the next layer in the network is a fully connected one with a size of 256 hidden neurons. But in or-

Table 1: Distribution of biased and unbiased articles ("a." for articles) in the training and validation data set.

	Training	Test
Total # samples	588837	147210
# samples w/ $l > 2500$	0.62%	0.58%
# samples w/ $l > 5000$	0.04%	0.03%
# of biased articles	290513	72629
# non-biased articles	298324	74581
% biased articles	49.34	49.34
% of non-biased articles	50.66	50.66

der for the convolutional output to serve as an input to this layer, it needs to be reduced in dimensionality. The way we chose to do that was using the flatten method, which keeps all of the values but flatten them in a long vector. A *ReLU* activation function and a dropout layer with a rate of 0.5 were used here.

Output layer. The last layer is a fully connected layer with one neuron. The sigmoid activation function is used to provide a binary output.

2.3 Architecture Variations

We developed and evaluated the following architecture variations:

1. The first variation only keeps the most important features by using the *GlobalMaxPooling* method. Keeping only the max values has shown good results when used with convolutional layers (Scherer et al., 2010).

2. The second variation uses the *flatten* method, which also transforms the convolutional filters weights to a one-dimensional vector, but does that by keeping all of the values (text features) from the convolutional filters and concatenates them in the resulting vector. Obviously, the length of the output from this method will be greater, usually much greater, than the previous method.

3 Evaluation

3.1 Data Set

Note that the actual SemEval 2019 Task 4 test data set is hidden and only used for submission. Thus, we used the training and validation data set of the SemEval 2019 Task 4 data set for training and testing our models before the SemEval submissions. Note also that, the biased news articles in the training and test data set always originate from sources other than the non-biased news articles. Biased news articles originated mainly from `foxbusiness.com`, `counterpunch.org`, `motherjones.com`, `truthdig.com`, and `dailywire.com`, while unbiased news articles originated mainly from `abqjournal.com` and `apnews.com`. The SemEval 2019 Task 4 is, thus, to some degree artificial, as, in reality, the source of a given article could be used as a feature for the classification.

Due to this correlation between the article's bias and its publisher, to have a generic model as much as possible it was important to have articles from the same publisher present in both sets. We decided to merge the training and validation data set, to shuffle the data randomly, and to split it into a training, validation, and testing data set by 60:20:20. Table 1 shows basic statistics about the used training data set and test data set.

Table 2: Hyperparameters.

Parameter	Value
CNN filter size for CNN	64
CNN kernel size for CNN	4
MaxPooling1D pool size	4
Dropout rate	0.2
Dense layer units	256
Layers Activation function	ReLu
Optimizer	Adam
Learning Rate	Adaptive (0.001→0.00001)
Loss function	Binary crossentropy
Batch Size	32

3.2 Evaluation Settings

We developed our models using Keras v2.1.2 with a Tensorflow v1.0.0 backend. Training the model was performed on a machine with 64GB memory and a GeForce GTX 1080 Ti GPU.

We implemented and evaluated our basic model using several word-based embedding methods, where an embedding vector is generated for each unique word on the text corpus. These embeddings can be categorized into two main categories: (1) pretrained word vectors and (2) custom word vectors (here, trained on the articles' content).

We fine-tuned the hyperparameters of our basic model using the dedicated validation data set. In the end, we used the parameters as shown in Table 2. Note that these optimal hyperparameters showed to be the best-performing ones on both Google's prebuilt word2vec and the custom word2vec which we had trained on our training data.

Table 3: Evaluation results on the custom validation split using various embedding methods.

Embedding	# Dim.	Accuracy	Precision	Recall	F1
Google's word2vec (general)	300	0.9255	0.9295	0.9197	0.9246
Stanford's GloVe (general)	100	0.9198	0.9161	0.9216	0.9188
Facebook's fastText (general)	300	0.9277	0.9378	0.9021	0.9196
Custom word2vec	100	0.9234	0.9246	0.9197	0.9222
Custom fastText	100	0.9114	0.9309	0.8860	0.9079
Custom news word2vec	100	0.9123	0.9142	0.9073	0.9108

Table 4: Results for the adapted CNN architecture.

Architecture	Accuracy
Model using GlobalMaxPooling	0.9225
Model using Flatten	0.9255

3.3 Evaluation Results

3.3.1 Evaluating the Basic Architecture

Table 3 presents the evaluation results for all used embedding methods. The embedding models gave similar results, with some slight differences in the model accuracies for some of them. Thus, we decided to go with one of the better performing models for the final SemEval test runs (see Sec. 3.3.3).

Although the fastText word embeddings are said to perform as well as word2vec embeddings while being trained much faster (Grave et al., 2017) and although the domain-specific pre-trained word2vec embeddings are said to perform better than the general pre-trained word2vec embeddings (Kim, 2014), the general pre-trained word2vec embeddings lead to the best results in our evaluation. A reason for that could be that the words used in the news articles are more spread in terms of the domains to which they belong. Thus, the word vectors trained by Google on 100 billion words of Google News performed even better than a specific word2vec model trained on millions of news articles.

3.3.2 Evaluating the Architecture Variations

We trained the extended models with the same hyperparameters as our basic model and used Google's word2vec as word embedding method based on our results from Sec. 3.3.1. The evaluation results are shown in Table 4.

Even though the training times for the flatten method were much longer due to the huge difference in terms of the number of trainable parameters, we obtained slightly better results than the GlobalMaxPooling method. Thus, the model chosen for the SemEval submission was the model employing the flattening method.

3.3.3 Evaluating on SemEval's Test Data Set

Applying our basic model – using our second architecture variation, Google's word2vec embedding model, the hyperparameters shown in Table 2 and the model itself trained using our mixed, "artificial" data sets – on the official SemEval test data set via official approach submissions, we obtained accuracy of 0.602, a precision of 0.560, a recall of 0.955, and a F1 score of 0.706. It becomes apparent that the precision value is considerably low, while the recall rate is the highest achieved rate among all submitted systems. Based on our investigations so far, we believe that one reason for the low accuracy on the evaluation data set is the labeling of the used data sets: while the train and test data set's labels depend solely on the article's publisher, the labels of the evaluation data set were hand-labeled on a single article basis.

4 Conclusion

In this paper, we presented a convolutional neural network architecture for determining whether a given news articles is *biased* (i.e., *hyperpartisan*) or not. In our experiments, we found that a convolutional neural network containing the flatten function and using Google's word2vec embeddings performs best. In the official SemEval 2019 Task 4 test runs, we obtained comparably low precision and accuracy values, while gaining the highest recall rate among all 42 participating teams. For the future, besides evaluating a deeper CNN, we plan to develop two further approaches. The first one will be based on LSTMs. The second model will be a hybrid model, consisting of a CNN layer, which is supposed to learn the text features, appended by an LSTM layer for learning sequence patterns of those features. Furthermore, we plan to work on a non-deep learning approach which assigns controversy scores to news articles and, in this way, determines the bias of the articles.

References

Ramy Baly, Georgi Karadzhov, Dimitar Alexandrov, James R. Glass, and Preslav Nakov. 2018. Predicting Factuality of Reporting and Bias of News Media Sources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3528–3539.

Eric Baumer, Elisha Elovic, Ying Qin, Francesca Polletta, and Geri Gay. 2015. Testing and Comparing Computational Approaches for Identifying the Language of Framing in Political News. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT'15, pages 1472–1482.

Edouard Grave, Tomas Mikolov, Armand Joulin, and Piotr Bojanowski. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, EACL'17, pages 427–431.

Felix Hamborg, Karsten Donnay, and Bela Gipp. 2018. Automated identification of media bias in news articles: an interdisciplinary literature review. *International Journal on Digital Libraries*, pages 1–25.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP'14, pages 1746–1751.

Craig Macdonald, Iadh Ounis, and Ian Soboroff. 2007. Overview of the TREC 2007 Blog Track. In *Proceedings of The Sixteenth Text REtrieval Conference*, TREC'07.

François Mairesse, Marilyn A. Walker, Matthias R. Mehl, and Roger K. Moore. 2007. Using linguistic cues for the automatic recognition of personality in conversation and text. *J. Artif. Intell. Res.*, 30:457–500.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. SemEval-2016 Task 4: Sentiment Analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval@NAACL-HLT 2016, pages 1–18.

Iadh Ounis, Craig Macdonald, Maarten de Rijke, Gilad Mishne, and Ian Soboroff. 2006. Overview of the TREC 2006 Blog Track. In *Proceedings of the Fifteenth Text REtrieval Conference*, TREC 2006.

Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: the detection, classification and structure of arguments in text. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law*, pages 98–107.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2016. Credibility Assessment of Textual Claims on the Web. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*, CIKM 2016, pages 2173–2178.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, ACL'18, pages 231–240.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic Models for Analyzing and Detecting Biased Language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL'13, pages 1650–1659.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval@ACL'17, pages 502–518.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. SemEval-2015 Task 10: Sentiment Analysis in Twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, SemEval@NAACL-HLT 2015, pages 451–463.

Dominik Scherer, Andreas C. Müller, and Sven Behnke. 2010. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In *Proceedings of the International Conference on Artificial Neural Networks*, ICANN'10, pages 92–101.

Team Peter-Parker at SemEval-2019 Task 4: BERT-Based Method in Hyperpartisan News Detection

Zhiyuan Ning, Yuanzhen Lin, Ruichao Zhong
Department of Computer Science and Technology
Beijing Normal University - Hong Kong Baptist University United International College
Zhuhai, P.R.China
{chineseperson5, jeremy0077jj, answer980810}@gmail.com

Abstract

This paper describes the team peter-parker's participation in Hyperpartisan News Detection task (SemEval-2019 Task 4), which requires to classify whether a given news article is bias or not. We decided to use JAVA to do the article parser and the BERT model to do the bias analysis and prediction. Furthermore, we will show experiment results with analysis.

1 Introduction

As the Hyperpartisan News Detection is getting more and more popular in NLP area in recent years, our team decided to focus on such kind of topic and choose task 4 in 2019 SemEval competition, which requires to decide whether a given news article is showing an unreasoning or blind allegiance to some specific groups or persons(Kiesel et al., 2019). For the task, it also requires the competitor's model to classify the news article in one of the two classes, bias or not. In previous SemEval competition, the classification tasks were mostly regarded as sentiment analysis on Twitter, news or scientific paper and so on. For SemEval 2019, the task focus on bias detection, which gives great help for people in daily life to acquire the news and articles in a more objective way.

So far, the machine learning approaches to do the bias detection were mostly using the RNN(Iyyer et al., 2014) or the Word Vectors(Anil Patankar and Bose, 2017), which can get the accuracy for more than 70%.

To reach a greater performance, we decided to adopt a state-of-the-art language model, BERT (Devlin et al., 2018), which set new records on many NLP tasks recently, into our political bias task analysis.

For dealing with the given large dataset, JAVA was used as a parser tool to help us make those training articles more readable.

2 Model Description

Our task is to predict whether an article or news is bias or not, which is entirely a binary classification task. Recently, Google released an essay about BERT and its code. They also provided the performance of BERT model on different tasks in the essay. One of them is similar to our task, which is called SST-2. It is a binary classification task, which is The Stanford Sentiment Treebank, a binary single-sentence classification task consisting of sentences extracted from movie reviews with human annotations of their sentiment. According to the result (about 95% accuracy), BERT perform pretty well on SST-2 task.

We chose to use BERT to do the task since it had better result on binary classification. We will introduce the BERT model and its detailed implementation in this section. In the following part, we will introduce the model architecture, input representation, pre-training procedure, and fine-tuning procedure.

2.1 Model Architecture

The architecture of BERT is a multi-layer bidirectional Transformer encoder. In this model, it indicates the number of layers as L, the hidden size as H, and the number of self-attention heads as A. It set the feed-forward/filter size to be 4H. It also provides two model sizes.

- BERT-Base: L=12, H=768, A=12, Total Parameters=110M
- BERT-Large: L=24, H=1024, A=16, Total Parameters=340M

2.2 Input Representation

No matter the input in one token sequence is a single text sentence or a pair of text sentences, BERT input representation is capable of representing them. To construct the input representation of a given token, we merged the corresponding token, segment and position embeddings.

2.3 Google pre-trained BERT

Instead of using traditional left-to-right or right-to-left language models, Google pre-train BERT using two new unsupervised prediction tasks which are Masked LM and Next Sentence Prediction.

2.4 Fine-tuning Procedure

It is easy to do BERT fine-tuning for sequence-level classification tasks. By construction corresponds to the special [CLS] word embedding, we take the final hidden state (i.e., the output of the Transformer) for the first token in the input for the interest of obtaining a fixed-dimensional pooled representation of the input sequence. We denote this vector as C $\in R^H$. Additionally, the only new parameters added during fine-tuning are for a classification layer W $\in R^{K*H}$. (K is the number of classifier labels). The label probabilities P$\in R^K$ are computed using a standard softmax, P = softmax(CW^T). All the parameters of BERT and W are fine-tuned cooperatively to maximize the log-probability of the correct label. Some modification must be done slightly on the above procedure in a particular task manner for span-level and token-level prediction tasks.

3 Experiments

3.1 Parse on the Datasets

A good training data cannot be made without data cleaning. There were 600,000 training data (by publisher) and 150,000 validation data (by publisher) and 645 training data (by article). For such a huge dataset, we first split data into 75 separated files, each contained 10,000 news articles so that they were easy to be opened by text editors.

After doing this, we used JAVA to do data parsing and cleaning and BERT model to do bias analysis and prediction. In the articles, some of the characters are escaped as HTML such as that " & " became &amp. It was really easy to unescape them with Java, which involved only one line of method invocation: StringEscapeUtils.unescapeHtml4().

There were some unknown Unicode characters in the articles, so it is good to be removed. Unfortunately, when applying some regular expressions to the articles to remove those characters, some of the articles in other languages would be gone, for example, Chinese, since it was hard to find all the occurring unknown characters in all news articles and we had to use a simple method which was blindly removing all the words, not in the range of \x00 to \x7F in Unicode. So, all the unknown characters could not be removed in order for retaining meaningful words in other languages.

Another problem was that most of the articles contained too many urls and a number of html tags because these articles are parsed from the internet. These might affect the performance and accuracy, so we used a set of regular expressions to catch and remove all the urls and html tags in different forms. Finally, another regular expression was applied to remove the duplicate punctuation such as a line of only periods to divide the articles.

3.2 Models and Training

After cleaning the data, we used BERT to train it with two procedures, which are pre-training and fine-tuning procedures.

3.2.1 Pre-training Procedure

According to Google, BERT needs plenty of time and resources to finish the pre-training procedure. Google used 4 Cloud TPUs in Pod configuration (16 TPU chips total) to train BERT-Base model. Considering the limited time and resource, we decided to download the pre-training model from Google instead of training it by ourselves. For the reason that the data consisted of different languages and Google only released the base model for multilingual data, we could only choose this, which is shown below:

- BERT-Base, Multilingual Cased: 104languages, L=12, H=768, A=12, Total Parameters=110M

3.2.2 Fine-tuning procedure

3.2.2.1 Modify processor

BERT needs an explicit input to train or predict, and it contains the processor to process the input of the model. For our task, we created a new processor for the dataset.

For a model that needs to perform a complete process of training, cross-validation, and testing, our custom processor needed to inherits the DataProcessor, overloads the get_labels function to gain label and also overloads the get_train_examples function, get_dev_examples function and get_test_examples function to get the individual input. These are invoked in the main function flags.do_train, flags.do_eval, and flags.do_predict phases, respectively. The contents of the three functions are much the same, except that you specify the address of the file to be read into.

Modifying get_train_examples function, the function needed to return a list which is made up by InputExample class. The InputExample class only contained the initialization functions. The initialization function only needed the variable guid, which was used to distinguish each example and it could be defined as train-%d'%(i) way. text_a is one string, text_b is another string. Text_a and text_b are two strings and they were merged with [CLS] and [SEP] to become [CLS]text_a[SEP]text_b[SEP]. This merged string was then given to the model. The last parameter, label, was also a string, and it should be guaranteed to appear in the list returned by the get_labels function.

Functions get_dev_examples and get_test_examples were modified using the same method above.

3.3 Results

The prediction procedure was done with the TIRA(Potthast et al., 2019) machine provided by the organizer.

Prediction on Training Set

We split the training set to get the validation set and the test set. The first experiment contained 10000 training articles, 2000 validation articles, 2000 test articles. Also, the proportion of the five categories was equal in the three sets mention above. All the articles are received from them in order from the given training set. Figure 1 shows the result of our first experiments.

Prediction on Test Set

In the second experiment, we directly used the aforementioned model to predict the given test set (byarticle) since the result was so good in the first experiment. But the accuracy was dropped down to 0.6077. The results is shown on the figure 2.

Prediction on Final Test Set

The accuracy we obtained in the second experiment was so bad and we thought that it was due to a lack of training data. So, we anticipated that more training data would help. Next, we trained all the data and predicted the test set pan19-hyperpartisan-news-detectionby-article-test-dataset-2018-12-07. The result was even worse in this final experiment, which is shown in the figure 3. The accuracy dropped to the lowest point, 0.5031. It was just like random guessing.

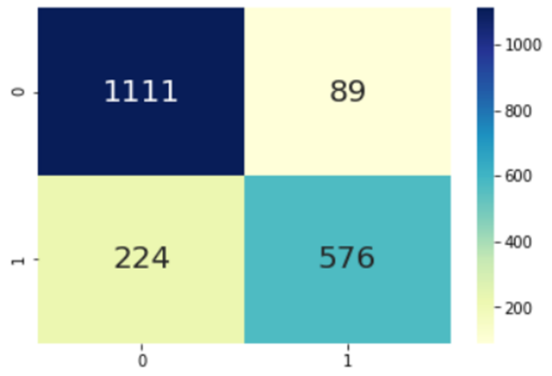


Figure 1: First experiment accuracy:0.9125, 0 is not bias,1 is bias.

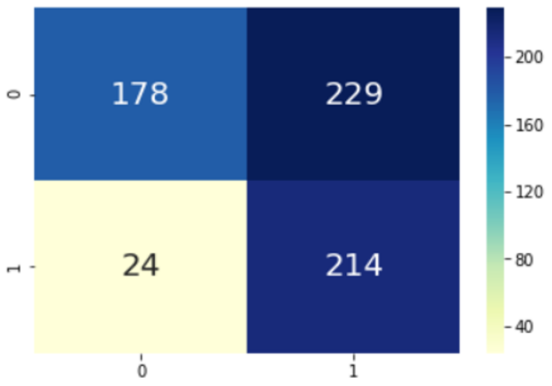


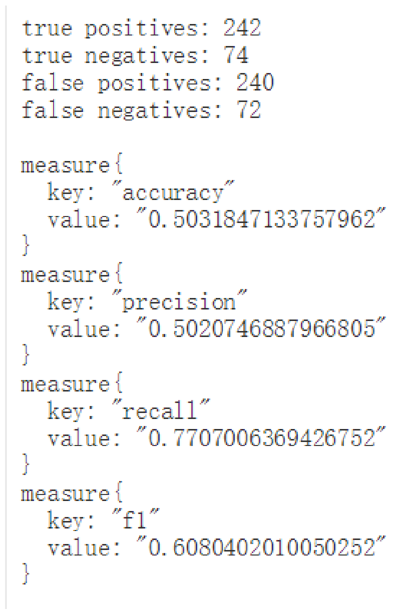
Figure 2: Second experiment accuracy:0.6077, 0 is not bias,1 is bias.

3.4 Error Analysis

We guess the reason why we get two different accuracies in two kinds of articles (by publisher, by article), is that the source of these two articles are different. Data by publisher is decided by the press and publisher, and the other is decided by manual selection. We only trained the data by publisher, so, it seems that it will not perform well on the type of data by article.

4 Conclusion

It was a great experience to use BERT to do the hyperpartisan news detection task although the result was not quite promising compared with the



```
true positives: 242
true negatives: 74
false positives: 240
false negatives: 72

measure{
  key: "accuracy"
  value: "0.5031847133757962"
}
measure{
  key: "precision"
  value: "0.5020746887966805"
}
measure{
  key: "recall"
  value: "0.7707006369426752"
}
measure{
  key: "f1"
  value: "0.6080402010050252"
}
```

Figure 3: Final experiment accuracy:0.5031.

one that Google has achieved. There are many works we can improve in this task, for example, we may do text summarization before training the data since every article is very long. All in all, more effort still need to be spent in the future time.

References

Anish Anil Patankar and Joy Bose. 2017. Bias discovery in news articles using word vectors. pages 785–788.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Mohit Iyyer, Peter Enns, Jordan L. Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *ACL*.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. TIRA Integrated Research Architecture. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer.

Team QCRI-MIT at SemEval-2019 Task 4: Propaganda Analysis Meets Hyperpartisan News Detection

Abdelrhman Saleh[1], **Ramy Baly**[2], **Alberto Barrón-Cedeño**[3], **Giovanni Da San Martino**[3], **Mitra Mohtarami**[2], **Preslav Nakov**[3], **James Glass**[2]

[1]Harvard University, MA, USA

[2]MIT Computer Science and Artificial Intelligence Laboratory, MA, USA

[3]Qatar Computing Research Institute, HBKU, Qatar

abdelrhman_saleh@college.harvard.edu,
{baly, mitram, glass}@mit.edu
{albarron, gmartino, pnakov}@hbku.edu.qa

Abstract

We describe our submission to SemEval-2019 Task 4 on Hyperpartisan News Detection. We rely on a variety of engineered features originally used to detect propaganda. This is based on the assumption that biased messages are propagandistic and promote a particular political cause or viewpoint. In particular, we trained a logistic regression model with features ranging from simple bag of words to vocabulary richness and text readability. Our system achieved 72.9% accuracy on the manually annotated testset, and 60.8% on the test data that was obtained with distant supervision. Additional experiments showed that significant performance gains can be achieved with better feature pre-processing.[1]

1 Introduction

The rise of social media has enabled people to easily share information with a large audience without regulations or quality control. This has allowed malicious users to spread disinformation and misinformation (a.k.a. "fake news") at an unprecedented rate. Fake news is typically charactterized as being hyperpartisan (one-sided), emotional and riddled with lies (Potthast et al., 2018). The SemEval-2019 Task 4 on Hyperpartisan News Detection (Kiesel et al., 2019) focused on the challenge of automatically identifying whether a text is hyperpartisan or not.

While hyperpartisanship is defined as "exhibiting one or more of blind, prejudiced, or unreasoning allegiance to one party, faction, cause, or person", we model this task as a binary document classification problem. Scholars have argued that all biased messages can be considered propagandistic, regardless of whether the bias was intentional or not (Ellul, 1965, p. XV).

[1]Our system is available at https://github.com/AbdulSaleh/QCRI-MIT-SemEval2019-Task4

Thus, we approached the task departing from an existing model for propaganda identification (Barrón-Cedeño et al., 2019). Our hypothesis is that propaganda is inherent in hyperpartisanship and that the two problems are two sides of the same coin, and thus solving one of them would help solve the other. Our system consists of a logistic regression model that is trained with a variety of engineered features that range from word and character TF.IDF n-grams and lexicon-based features to more sophisticated features that represent different aspects of the article's text such vocabulary richness and language complexity.

Our official submission achieved an accuracy of 72.9% (while the winning system achieved 82.2%). This was achieved using word and character n-grams. Moreover, post-submission experiments have shown that further performance improvements can be achieved by carefully preprocessing the engineered features.

2 Related Work

The analysis of bias and disinformation has attracted significant attention, especially after the 2016 US presidential election (Brill, 2001; Finberg et al., 2002; Castillo et al., 2011; Baly et al., 2018a; Kulkarni et al., 2018; Mihaylov et al., 2018; Baly et al., 2019). Most approaches have focused on predicting credibility, bias or stance.

Stance detection was considered as an intermediate step for detecting fake claims, where the veracity of a claim is checked by aggregating the stances of the retrieved relevant articles (Baly et al., 2018b; Nakov et al., 2019). Several stance detection models have been proposed including deep convolutional neural networks (Baird et al., 2017), multi-layer perceptrons (Hanselowski et al., 2018), and end-to-end memory networks (Mohtarami et al., 2018).

The stylometric analysis model of Koppel et al. (2007) was used by Potthast et al. (2018) to address hyperpartisanship. They used articles from nine news sources whose factuality has been manually verified by professional journalists. Writing style and complexity were also considered by Horne and Adal (2017) to differentiate real news from fake news and satire. They used features such as the number of occurrences of different part-of-speech tags, swearing and slang words, stop words, punctuation, and negation as stylistic markers. They also used a number of readability measures. Rashkin et al. (2017) focused on a multi-class setting (real news vs. satire vs. hoax vs. propaganda) and relied on word n-grams.

Similarly to Potthast et al. (2018), we believe that there is an inherent style in propaganda, regardless of the source publishing it. Many stylistic features were proposed for authorship identification, i.e., the task of predicting whether a piece of text has been written by a particular author. One of the most successful representations for such a task are character-level n-grams (Stamatatos, 2009), and they turn out to represent some of our most important stylistic features.

More details about research on fact-checking and the spread of fake news online can be found in recent surveys (Lazer et al., 2018; Vosoughi et al., 2018; Thorne and Vlachos, 2018).

3 System Description

We developed our system for detecting hyperpartisanship in news articles by training a logistic regression classifier using features such as character and word n-grams, lexicon-based indicators, and readability and vocabulary richness measures. Below, we describe these features in detail.

Character 3-grams. Stamatatos (2009) argued that, for tasks where the topic is irrelevant, character-level representations are more sensitive than token-level ones. We hypothesize that this applies to hyperpartisan news detection, since articles on both sides of the political spectrum may be discussing the same topics. Stamatatos (2009) found that "the most frequent character n-grams are the most important features for stylistic purposes". These features capture different style markers, such as prefixes, suffixes and punctuation marks. Following the analysis in Barrón-Cedeño et al. (2019), we include TF.IDF-weighted character 3-grams in our feature set.

Word n-grams Bag-of-words (BoW) features are widely used for text classification. We extracted the k most frequent $[1, 2]$-grams, and we represented them using their TF.IDF scores. We ignored n-grams that appeared in more than 90% of the documents, most of which contained stopwords and were irrelevant with respect to hyperpartisanship. Furthermore, we incorporated Naive Bayes by weighing the n-grams based on their importance for classification, as proposed by Wang and Manning (2012). We define $\boldsymbol{x}_i \in \mathbb{R}^{|V|}$ as a row vector in the TF.IDF feature matrix, representing the i^{th} training sample with a target label $y_i \in \{0, 1\}$, where V is the vocabulary size. We also define vectors $\boldsymbol{p} = \alpha + \sum_{i:y_i=1} \boldsymbol{x}_i$ and $\boldsymbol{q} = \alpha + \sum_{i:y_i=0} \boldsymbol{x}_i$, and we set the smoothing parameter α to 1. Finally, we calculate the vector:

$$\boldsymbol{r} = \log\left(\frac{\boldsymbol{p}/\parallel \boldsymbol{p} \parallel}{\boldsymbol{q}/\parallel \boldsymbol{q} \parallel}\right) \tag{1}$$

which is used to scale the TF.IDF features to create the NB-TF.IDF features as follows:

$$\boldsymbol{x}_i' = \boldsymbol{r} \circ \boldsymbol{x}_i, \quad \forall i \tag{2}$$

Bias Analysis We analyze the bias in the language used in the documents by (*i*) creating bias lexicons that contain *left* and *right* bias cues, and (*ii*) using these lexicons to compute two scores for each document, indicating the intensity of bias towards each ideology. To generate the list of cues that signal biased language, we use Semantic Orientation (SO) (Turney, 2002) to identify the words that are strongly associated with each of the left and right documents in the training dataset. Those SO values can be either positive or negative, indicating association with right or left biases, respectively. Then, we select words whose absolute SO value is $\geq$ 0.4 to create two bias lexicons: BL_{left} and BL_{right}. Finally, we use these lexicons to compute two bias scores per document according to Equation (3), where for each document D_j, the frequency of cues in the lexicon BL_i that are present in D_j is normalized by the total number of words in D_j:

$$bias_i(D_j) = \frac{\displaystyle\sum_{cue \in BL_i} count(cue, D_j)}{\displaystyle\sum_{w_k \in D_j} count(w_k, D_j)} \tag{3}$$

Lexicon-based Features. Rashkin et al. (2017) studied the occurrence of specific types of words in different kinds of articles, and showed that words from certain lexicons (e.g., negation and swear words) appear more frequently in propaganda, satire, and hoax articles than in trustworthy articles. We capture this by extracting features that reflect the frequency of words from particular lexicons. We use 18 lexicons from Wiktionary, Linguistic Inquiry and Word Count (LIWC) (Pennebaker et al., 2001), Wilson's subjectives (Wilson et al., 2005), Hyland's hedges (Hyland, 2015), and Hooper's assertives (Hooper, 1975). For each lexicon, we count the total number of words in the article that appear in the lexicon. This resulted in 18 features, one for each lexicon.

Vocabulary Richness Potthast et al. (2018) showed that hyperpartisan outlets tend to use a writing style that is different from mainstream outlets. Different topic-independent features have been proposed to characterize the vocabulary richness, style and complexity of a text. For this task, we used the following vocabulary richness features: (*i*) type–token ratio (*TTR*), or the ratio of types to tokens in the text, (*ii*) *Hapax Legomena*, or the number of word types appearing only once in the text, (*iii*) *Hapax Dislegomena*, or the number of types appearing twice in the text, (*iv*) *Honore's R*, which is calculated as a combination of types, tokens, and hapax legomena (Honore, 1979):

$$\text{Honore's R} = \frac{100 \times \log(|\text{tokens}|)}{1 - |\text{Legomena}|/|\text{types}|} \tag{4}$$

We further used (*v*) *Yule's characteristic K*, which is defined as the chance of a word occurring in a text, estimated as following a Poisson distribution (Yule, 1944):

$$\text{Yule's K} = 10^4 \cdot \frac{\sum_i i^2|\text{types}_i| - |\text{tokens}|}{|\text{tokens}|^2}, \tag{5}$$

where tokens refer to all words in a text (including repetitions), types refer to distinct words, i are the tokens' frequency ranks (1 being the least frequent), and types$_i$ are the number of tokens with the i^{th} frequency.

Readability We also used the following readability features, which were originally designed to estimate the level of text complexity: (*i*) *Flesch–Kincaid grade level* represents the US grade level necessary to understand a text (Kincaid et al., 1975), (*ii*) *Flesch reading ease* is a score for measuring how difficult a text is to read (Kincaid et al., 1975), and (*iii*) *Gunning fog index* estimates the years of formal education necessary to understand a text (Gunning, 1968).

4 Experiments and Results

4.1 Dataset

We trained our models on the Hyperpartisan News Dataset from SemEval-2019 Task 4 (Kiesel et al., 2019), which is split by the task organizers into

(*i*) *Labeled by-Publisher*, with 750K articles labeled via distant supervision, i.e., using labels for their publisher.[2] The labels are evenly distributed between "hyperpartisan" and "not-hyperpartisan." This set is further split into 600K articles for training and 150K for validation.

(*ii*) *Labeled by-Article:* This set contains 645 articles labeled using crowd-sourcing (37% are hyperpartisan and 63% are not). Only articles with a consensus among the annotators were included.

4.2 Experimental Settings

We trained a logistic regression (LR) model with a Stochastic Average Gradient solver (Schmidt et al., 2017) due to the large size of the dataset. In order to reduce overfitting, we used $\mathbb{L}_2$ regularization (with $C = 1$ as the regularization parameter). Moreover, feature normalization was needed since the different features represent different aspects of the text, and thus have very different scales. We tried to normalize each feature set by subtracting the mean and then scaling it to unit variance. However, we found that multiplying the features by constant scaling factors resulted in better performance. The scaling factor for each family of features was a hyperparameter that we tuned on the validation dataset.

We trained the classifier using the 600K training examples annotated *by-Publisher*, then we used the remaining 150K examples for evaluation. We fine-tuned the hyperparameters on the 645 *by-Article* examples.

[2]The publisher's labels are identified by BuzzFeed journalists or by the Media Bias/Fact Check project

	Features	Labeled by-**Article**				Labeled by-**Publisher**			
		Accuracy	Prec.	Rec.	F1	Accuracy	Prec.	Rec.	F1
1	BoW (TF.IDF)	67.8	53.8	89.1	67.1	56.7	55.1	72.5	62.6
2	BoW (NB-TF.IDF)	69.6	56.1	80.7	66.2	57.1	56.4	61.9	59.0
3	↳+ Char trigrams	74.0	62.5	73.5	67.6	54.8	54.3	60.8	57.4
4	↳+ Bias	75.2	67.7	62.6	65.1	54.5	55.0	50.4	52.6
5	↳+ Lexical	75.2	67.0	64.7	65.8	52.3	52.3	51.5	51.9
6	↳+ Vocab. Richness	75.8	67.1	67.6	67.4	50.9	50.8	52.5	51.7
7	↳+ Readability	76.0	66.4	70.6	68.4	51.6	51.5	53.9	52.7

Table 1: An incremental analysis showing the performance of different feature combinations, evaluated on the validation datasets labeled by *article* and by *publisher*.

The hyper-parameters include the number of most frequent word n-grams k, $k \in [50, 200, 700]^{\times 10^3}$, and the scaling parameters of the features, except for the n-grams. Eventually, we set $k = 200,000$, and we used the most-frequent word $[1, 2]$-grams. Moreover, we assessed the different feature sets, described in Section 3 by incrementally adding each set, one at a time, to the mix of all features.

4.3 Results

Table 1 illustrates the results obtained on both the *by-Article* set (which we used to fine-tune the model's hyper-parameters) and the *by-Publisher* set (which we used for evaluation). Our results suggest that scaling the TF.IDF values through Naive Bayes is better than using raw TF.IDF scores. Hence, this is what we used in subsequent experiments. We can also see that adding each group of features introduces a consistent improvement in accuracy on the *by-Article* data. However, we observed an opposite behaviour on the *by-Publisher* data. We believe this is due to the significant amount of noisy labels introduced by the distant supervision labeling strategy. Therefore, we based our decisions on the results obtained on the *by-Article* data since its labels are more accurate.

The normalization strategy, i.e., scaling the features using calibrated scaling parameters, yielded significant performance improvements. Unfortunately, we could not perform this by the competition deadline, and thus we submitted the system that was available at that time, which was based on the BoW (NB-TF.IDF) and character 3-gram features, as shown in row 3 in Table 1. Our system achieved 72.9% accuracy on the test *by-Article* data, ranking 20th/42, and 60.8% accuracy on the test *by-Publisher* data, ranking 15th/42.

5 Conclusion

We presented our submission to SemEval-2019 Task 4 on Hyperpartisan News Detection. We trained a logistic regression model with a feature set that included word and character n-grams, weighted using TF.IDF, after scaling using Naive Bayes. Our system achieved accuracy of 72.9% and 60.8% on the test datasets that were labeled *by-Article* and *by-Publisher*, respectively.

We further experimented with additional features that represent different aspects of the article's text such as its vocabulary richness, the kind of language it uses according to different lexicons, and its level of complexity. Initial experiments showed that these features hurt the model.

However, with proper pre-processing and scaling, we were able to achieve significant performance gains of up to 2% absolute in terms of accuracy. Unfortunately, we only obtained these results after the competition's deadline, and thus they were not considered as part of our submission. Yet, we have described them in order to facilitate further research.

Acknowledgments

This research is part of the Tanbih project,[3] which aims to limit the effect of "fake news", propaganda and media bias by making users aware of what they are reading. The project is developed in collaboration between the Qatar Computing Research Institute (QCRI), HBKU and the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL).

[3] http://tanbih.qcri.org/

References

Sean Baird, Doug Sibley, and Yuxi Pan. 2017. Talos targets disinformation with fake news challenge victory. https://blog.talosintelligence.com/2017/06/talos-fake-news-challenge.html.

Ramy Baly, Georgi Karadzhov, Dimitar Alexandrov, James Glass, and Preslav Nakov. 2018a. Predicting factuality of reporting and bias of news media sources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, EMNLP '18, pages 3528–3539, Brussels, Belgium.

Ramy Baly, Georgi Karadzhov, Abdelrhman Saleh, James Glass, and Preslav Nakov. 2019. Multi-task ordinal regression for jointly predicting the trustworthiness and the leading political ideology of news media. In *Proceedings of the 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '19, Minneapolis, MN, USA.

Ramy Baly, Mitra Mohtarami, James Glass, Lluís Màrquez, Alessandro Moschitti, and Preslav Nakov. 2018b. Integrating stance detection and fact checking in a unified corpus. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '18, pages 21–27, New Orleans, LA, USA.

Alberto Barrón-Cedeño, Giovanni Da San Martino, Israa Jaradat, and Preslav Nakov. 2019. Proppy: Organizing news coverage on the basis of their propagandistic content. *Information Processing and Management*.

Alberto Barrón-Cedeño, Giovanni Da San Martino, Israa Jaradat, and Preslav Nakov. 2019. Proppy: A system to unmask propaganda in online news. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, AAAI'19, Honolulu, HI, USA.

Ann M Brill. 2001. Online journalists embrace new marketing function. *Newspaper Research Journal*, 22(2):28.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on Twitter. In *Proceedings of the 20th International Conference on the World Wide Web*, WWW '11, pages 675–684, Hyderabad, India.

Jacques Ellul. 1965. *Propaganda: The Formation of Men's Attitudes*. Vintage Books, United States.

Howard Finberg, Martha L Stone, and Diane Lynch. 2002. Digital journalism credibility study. *Online News Association. Retrieved November*, 3:2003.

Robert Gunning. 1968. *The Technique of Clear Writing*. McGraw-Hill.

Andreas Hanselowski, Avinesh PVS, Benjamin Schiller, Felix Caspelherr, Debanjan Chaudhuri, Christian M. Meyer, and Iryna Gurevych. 2018. A retrospective analysis of the fake news challenge stance-detection task. In *Proceedings of the 27th International Conference on Computational Linguistics*, COLING '18, pages 1859–1874, Santa Fe, NM, USA.

Anthony Honore. 1979. Some Simple Measures of Richness of Vocabulary. *Association for Literary and Linguistic Computing Bulletin*, 7(2):172–177.

Joan B. Hooper. 1975. On assertive predicates. In J. Kimball, editor, *Syntax and Semantics*, volume 4, page 91124. Academic Press, New York.

Benjamin D Horne and Sibel Adal. 2017. This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. In *Proceedings of the International Workshop on News and Public Opinion at ICWSM*, Montreal, Canada.

Ken Hyland. 2015. *The International Encyclopedia of Language and Social Interaction*, chapter Metadiscourse. American Cancer Society.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, David Corney, Payam Adineh, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan news detection. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, SemEval '19, Minneapolis, MN, USA.

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. *Memphis TN Naval Air Station*, Research B.

Moshe Koppel, Jonathan Schler, and Elisheva Bonchek-Dokow. 2007. Measuring differentiability: Unmasking pseudonymous authors. *Journal of Machine Learning Research*, 8:1261–1276.

Vivek Kulkarni, Junting Ye, Steve Skiena, and William Yang Wang. 2018. Multi-view models for political ideology detection of news articles. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, EMNLP '18, pages 3518–3527, Brussels, Belgium.

David MJ Lazer, Matthew A Baum, Yochai Benkler, Adam J Berinsky, Kelly M Greenhill, Filippo Menczer, Miriam J Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild, et al. 2018. The science of fake news. *Science*, 359(6380):1094–1096.

Todor Mihaylov, Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Georgi Georgiev, and Ivan Koychev. 2018. The dark side of news community forums: Opinion manipulation trolls. *Internet Research*, 28(5):1292–1312.

Mitra Mohtarami, Ramy Baly, James Glass, Preslav Nakov, Lluís Màrquez, and Alessandro Moschitti. 2018. Automatic stance detection using end-to-end memory networks. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '18, pages 767–776, New Orleans, LA, USA.

Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, Pepa Gencheva, Georgi Karadzhov, Tsvetomila Mihaylova, Mitra Mohtarami, and James Glass. 2019. Automatic fact checking using context and discourse information. *ACM Journal of Data and Information Quality*.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: Liwc 2001. *LIWC Operators Manual 2001*.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A stylometric inquiry into hyperpartisan and fake news. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, ACL '18, pages 231–240, Melbourne, Australia.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, EMNLP '17, pages 2931–2937, Copenhagen, Denmark.

Mark Schmidt, Nicolas Le Roux, and Francis Bach. 2017. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112.

Efstathios Stamatatos. 2009. A Survey of Modern Authorship Attribution Methods. *Journal of the American Society for Information Science and Technology*, 60(3):538–556.

James Thorne and Andreas Vlachos. 2018. Automated fact checking: Task formulations, methods and future directions. In *Proceedings of the 27th International Conference on Computational Linguistics*, COLING '18, pages 3346–3359, Santa Fe, NM, USA.

Peter D. Turney. 2002. Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 417–424, Philadelphia, Pennsylvania.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science*, 359(6380):1146–1151.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, ACL '12, pages 90–94, Jeju Island, Korea.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT-EMNLP '05, pages 347–354, Vancouver, Canada.

George Udny Yule. 1944. *The Statistical Study of Literary Vocabulary*. Cambridge University Press.

Team Xenophilius Lovegood at SemEval-2019 Task 4: Hyperpartisanship Classification using Convolutional Neural Networks

Albin Zehe, Lena Hettinger, Stefan Ernst, Christian Hauptmann and Andreas Hotho
DMIR Group, University of Wuerzburg
`<surname>@informatik.uni-wuerzburg.de`

Abstract

This paper describes our system for the SemEval 2019 Task 4 on hyperpartisan news detection. We build on an existing deep learning approach for sentence classification based on a Convolutional Neural Network. Modifying the original model with additional layers to increase its expressiveness and finally building an ensemble of multiple versions of the model, we obtain an accuracy of $67.52\,\%$ and an F1 score of $73.78\,\%$ on the main test dataset. We also report on additional experiments incorporating handcrafted features into the CNN and using it as a feature extractor for a linear SVM.

1 Introduction

The goal of SemEval 2019 Task 4 is to determine whether a news article blindly follows a political argumentation or not, which is referred to as "hyperpartisan news" (Kiesel et al., 2019). Instead of predicting the exact political orientation, it focuses on whether an article is hyperpartisan in any way. This is a very topical issue since news are easily able to reach millions of people over the internet, and in recent years have been excessively used to influence the population, for example regarding elections. Specifically, one sided media coverage influences a lot of readers without their knowledge, demonstrating the necessity of automated detection of hyperpartisan news.

Approach In this work, we make use of deep learning models to address this task. We decided to adapt the sentence CNN proposed by Kim (2014), as it has been shown to be a strong baseline for text classification tasks. Since the model was originally designed for the classification of sentences, we had to make some modifications in order to deal with the longer texts provided in this task. While the very shallow model originally proposed by Kim (2014) is enough to adequately represent sentences, we found that it is not expressive enough to model entire news articles. Thus, we added a second convolutional layer and a batch normalization layer (Ioffe and Szegedy, 2015) to the model. Additionally, we specified a maximum length for articles, after which they are cut off. These modifications will be described in more detail in Section 3. We also experiment with including some hand-crafted features into the model in an attempt to improve the performance. Finally, we build an ensemble of multiple models to obtain our final results.

2 Feature Extraction

In order to train our models, we need to represent the input texts in some machine readable form. The data provided in the task contains multiple kinds of information that we use in different parts of our system, namely the CNN model and the hand-crafted features extracted to provide additional information.

2.1 CNN Model

On the one hand, our main CNN model is based purely on the text of the articles and requires little pre-processing. For this part, we built a specialized parser[1] to remove HTML tags and split the texts into tokens, which are then directly used as input to the CNN. We chose to allow some special characters like punctuation marks to support the model identifying different sentences. Contractions like *they're* or *he's* are split to obtain separate tokens which can then be mapped to existing ones.

An article is then represented as a sequence of one-hot vectors, where each dimension corresponds to a word in the vocabulary. This sequence is concatenated to form a matrix $M \in \mathbb{N}^{l \times v}$, where l is the length of the article and v is the vocabulary size.

[1] `https://github.com/o8Gravemind8o/nlp_tokenizer`

Word Embeddings As is common for NLP tasks, we use embedding vectors to represent the semantic meaning of tokens. Since the provided datasets are rather large and previous work has shown that domain specific word embeddings can greatly improve classifier performance compared to general embeddings (Hettinger et al., 2018), we train our embeddings on these datasets. More specifically, we use Word2Vec (Mikolov et al., 2013) as well as FastText (Bojanowski et al., 2017) to retrieve different embeddings and see how they perform in different approaches.

Dealing with Variable Article Length Due to CNNs not being able to process input of arbitrary length, we decided to represent articles with a fixed length of 2000 tokens. Articles that exceed this length are cut off at 2000 tokens. This saves a lot of training time and affects less than 3 % of the articles. In the same way, we pad articles shorter than 2000 tokens with zeros to achieve a consistent input size.

2.2 Hand-Crafted Features

On the other hand, we employ hand-crafted features partially based on the metadata that is contained in the HTML. We do this to enable our classifiers to use information that may not be contained in the raw text and also possibly recover information that is lost by the length limit we impose on the articles. For this purpose, we choose several kinds of information from the articles, inspired by Potthast et al. (2018). First, we count (a) every token in the article and (b) tokens which are placed between quotation marks. Furthermore, we use the corresponding HTML tag to count paragraphs and calculate the average number of tokens per paragraph. Finally, we use the overall number of hyperlinks as well as the number of internal and external links. These values are concatenated to form a feature vector that can be used as input to an SVM or as additional input to the CNN.

3 Model Architectures

In this section, we describe the architectures we evaluated in our experiments, starting with the base CNN model from Kim (2014) and extending this model step by step. We also describe an experiment to use the CNN model as a feature extractor for an SVM.

Base Model: Sentence CNN We use the sentence CNN from Kim (2014) as a starting point for our model, illustrated in Figure 1. Articles are fed into the CNN represented by the matrix described in Section 2.1. The first layer of the network then converts the one-hot representation to an embedding representation. To obtain a vector representation of the words, we use two different approaches described in Kim (2014), CNN-Rand and CNN-Static. With CNN-Rand, word vectors are initialized randomly and learned during training. CNN-Static uses the embeddings described in Section 2.1 and does not change them during training. The CNN extracts features from the articles through a convolution layer followed by max-over-time pooling. Classification of an article is obtained by flattening the max-pool feature map and passing the features through a fully connected layer.

First Extension: Article CNN As the base model discards all but one feature from the convolution activation map of each filter by using max-over-time pooling, a lot of features are lost. In the model's original task, which is sentence classification, this is not much of an issue. However, our experiments show that for the classification of articles (which are much longer than one sentence), we need to keep more information.

Therefore, we modify the sentence CNN and use this as a second model, which we call *Article CNN*. We add a second convolutional layer after the first one to learn features that cover a larger range in the article. Furthermore, we add a batch normalization layer after the first and after the second convolutional layer to speed up the training process (Ioffe and Szegedy, 2015).

Second Extension: Article CNN with Hand-Crafted Features In an attempt to incorporate additional information into our model, we provided the model with some handcrafted features described in Section 2.2. To make use of handcrafted features, we append them to the flattened output of the pooling layer of the Article CNN. Since we found that the model can not learn from the combination of CNN and hand-crafted features with only one dense layer at the end, a second dense layer is inserted before the first one. We refer to the resulting third model as *Article CNN HC*.

Model Variant: CNN as a Feature Extractor In an additional experiment, we use the CNN as a feature extractor for an SVM. To this end, we first

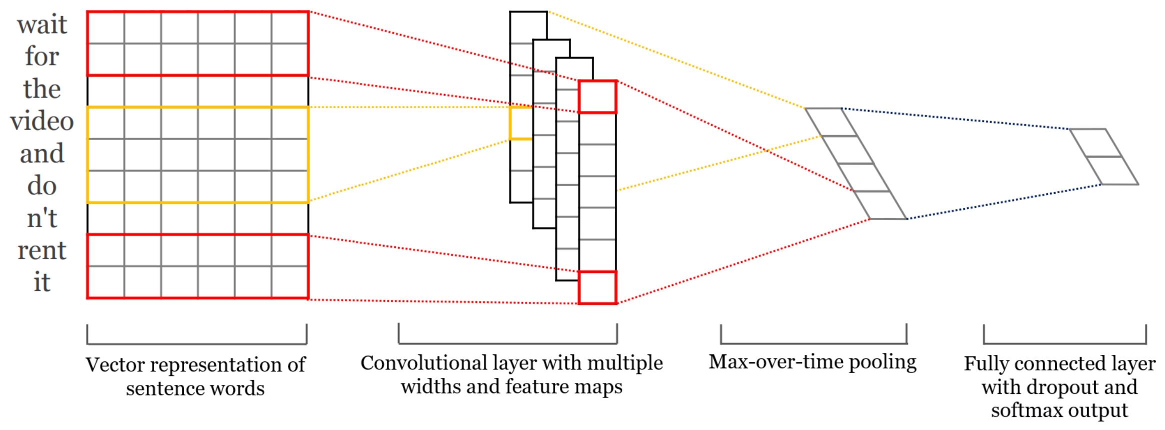

wait
for
the
video
and
do
n't
rent
it
Vector representation of sentence words
Convolutional layer with multiple widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 1: Architecture of sentence CNN by Kim (2014).

train the Article CNN regularly. We then convert an article to a feature representation by feeding it into the trained CNN and extracting the representation before the first dense layer. This vector is concatenated with the hand-crafted features described in Section 2.2 and used to train a linear Support Vector Machine (SVM) as an alternative to the neural classifier of the CNN.

4 Evaluation

After defining our models, we now shortly describe the datasets (for a more detailed description, see Kiesel et al. (2019)) before presenting the results of training and evaluation.

4.1 Data

For training and validation, the task provides 750 000 news articles, which are equally distributed into the two classes hyperpartisan and not hyperpartisan. 600 000 of these were used for training, the remaining 150 000 for validation. These articles have not been labeled individually, but according to their publisher, making them a form of weakly labelled data. The official evaluation was then performed on two concealed datasets, one with articles manually labeled by humans and one again labeled by publisher. The evaluation data contains 628 articles labeled individually and 4000 by publisher, with both being equally distributed into the two classes.

4.2 Metrics

We determine the performance of our models by measuring accuracy (Acc) and F1 score (F1). As accuracy is the official evaluation metric of SemEval Task 4, we optimize for this metric.

4.3 Training and Results

We chose to optimize the hyperparameters of our models by random search (Bergstra and Bengio, 2012). The hyperparameters with the best accuracy values of each architecture are shown in Table 1. All configurations use the CNN-static variant. Models were trained for a maximum of 5 epochs with a batch size of 256. We employed early stopping when the validation accuracy did not improve for 8 consecutive batches.

Results on the Validation Dataset First, we report the results obtained by our models on the validation dataset. All results on this dataset are shown in Table 2. The best configuration of the Sentence CNN achieves a maximum accuracy of 62.13 % and an F1 score of 70.47 %.

Our first extension, the Article CNN, increases the accuracy by 1.68 percentage points and F1 score by 6.09 percentage points. We attribute this to the increased model capacity, which enables the model to represent articles more adequately. With the max-pooling layer after the first convolution layer, the whole article is reduced to one value per convolution filter, which covers a maximum of 11 words (filter size is 11). The second convolution layer contains information of several outputs of the first layer, hence learning higher level features. As a result of that, less information about the article is lost by max-pooling.

Our second modification, the Article CNN HC, however, decreases the model's performance, as does using an SVM as a classifier instead of the

	Convolution 1		Convolution 2		Dense Layer		Miscellaneous	
Model	Filter Size	Filters	Filter Size	Filters	Dropout keep	Units	Activation	Embeddings
Sentence CNN	5	168	—	—	0.9	168	tanh	Word2Vec
Article CNN	9,10,11	306	7	199	0.837	4179	ReLU	FastText
Article CNN HC	7	405	9	210	0.079, 0.274	1890+7, 123	ReLU	Word2Vec

Table 1: Best hyperparameters obtained via random search. Article CNN HC has two dense layers, hence two numbers for dropout and units. We evaluated multiple filter sizes for Article CNN only, due to time constraints.

Model	F1	Acc
Sentence CNN	70.47	62.13
Article CNN	**76.56**	63.81
Article CNN HC	63.02	60.30
Article CNN + SVM	62.07	58.18
SVM HC	66.91	52.83
Ensemble 3	68.69	64.94
Ensemble 5	68.98	**66.01**

Table 2: Best results achieved on the validation dataset.

	Article		Publisher	
Model	F1	Acc	F1	Acc
Article CNN	70.26	64.81	67.81	**66.78**
Ensemble 3	72.05	65.29	**70.09**	66.08
Ensemble 5	**73.78**	**67.52**	69.85	66.28

Table 3: Results on the hidden test datasets.

final fully-connected part of the CNN (Article CNN + SVM). We also trained an SVM purely on the hand-crafted features for comparison (SVM HC), leading to a lower accuracy but higher F1 score than both variants of Article CNN HC.

Finally, we used an ensemble of multiple models for prediction. To this end, we used either 3 or 5 of our best individual models and combined their predictions by majority vote. Because of the non-deterministic nature of training (shuffling of the input data and random initialization of the network) (Reimers and Gurevych, 2017), multiple versions of the same model with similar accuracy may rely on different features and make different mistakes. Thus, a combined prediction by the best models can improve overall accuracy. This is confirmed by the results of Ensemble 3 and 5 presented in Table 2, leading us to submit these ensembles as our final systems. Our best model, Ensemble 5, combines 5 individual Article CNNs.

Results on the Test Dataset For the final evaluation, participants were able to submit two models for evaluation on the non-public test sets. We submitted our two ensemble models comprised of multiple instances of the Article CNN. Additionally we were able to evaluate our single best Article CNN on both test sets, giving us the evaluation results but not appearing on the scoreboard. Results on the test dataset are shown in Table 3. As on the validation dataset, the ensembles outperform the Article CNN in prediction accuracy on the main test set. Our best performing model, the Ensemble 5, reaches an accuracy of 67.52 % on the by-article test set and 66.28 % on the by-publisher test set. This corresponds to rank 25 of 42 on the Main Leaderboard (by-article) and rank 4 out of 28 on the by-publisher Leaderboard.[2]

5 Conclusion

In this paper, we have described our approach for SemEval Task 4 to detect hyperpartisanship in news articles. We trained a CNN for sentence classification and improved its performance by adding a second convolution layer and batch normalization. Moreover, we combined this model with handcrafted features. However, this did not lead to an improvement of classification performance, nor did the use of an SVM as an alternative classifier. Finally, an increase of accuracy was achieved by combining our best models for ensemble prediction. Through this approach, we obtained an accuracy of 67.52 % and F1 score of 73.78 % on the by-article test dataset as well as 66.28 % accuracy and 69.85 % F1 score on the by-publisher data set. For future research, possible modifications to our Article CNN that may bring further improvement are using more channels for the input or different filter sizes for max pooling. Apart from that, using Transformer models (Vaswani et al., 2017) could be rewarding, as they have become the standard for many tasks in Natural Language Processing.

[2]We would like to note that, by F1 score, we would rank 14 out of 42 in the Main Leaderboard (by-article).

References

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Lena Hettinger, Alexander Dallmann, Albin Zehe, Thomas Niebler, and Andreas Hotho. 2018. Claire at semeval-2018 task 7: Classification of relations using embeddings. In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, USA.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. Semeval-2019 task 4: Hyperpartisan news detection. *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A stylometric inquiry into hyperpartisan and fake news. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 231–240.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Team yeon-zi at SemEval-2019 Task 4: Hyperpartisan News Detection by De-noising Weakly-labeled Data

Nayeon Lee[*], **Zihan Liu**[*], **Pascale Fung**
Center for Artificial Intelligence Research (CAiRE)
Department of Electronic and Computer Engineering
Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong
`[nyleeaa, zliucr].connect.ust.hk, pascale@ece.ust.hk`

Abstract

This paper describes our system submitted to SemEval-2019 Task 4: Hyperpartisan News Detection. We focus on removing the inherent noise in the hyperpartisanship dataset from both data-level and model-level by leveraging semi-supervised pseudo-labels and the state-of-the-art BERT model. Our model achieves 75.8% accuracy in the final by-article dataset without ensemble learning.

1 Introduction

With the ever-growing usage of internet, the problem of fake news that spreads in a destructive speed has attracted many attention. Fake news is a kind of news that is typically inflammatory, extremely one-sided (hyper-partisan) or untruthful to mislead the public into having distorted belief.

Previous works attempted to solve fake news problem from various aspects, ranging from knowledge-based (Wu et al., 2014; Shi and Weninger, 2016; Lee et al., 2018) to style-based (Wang, 2017; Potthast et al., 2018). There are some publicly available fake news datasets, however, often too small in size to be suitable for neural approaches (Horne and Adali, 2017; Pérez-Rosas et al., 2017). Recently, the organizers of SemEval2019 Task 4 (Kiesel et al., 2019) have released large-scale dataset to address fake news detection as a hyper-partisan news detection problem. The task is to determine whether a given article is hyper-partisan (extremely right-wing or left-wing) or not (mainstream). Such task will allow for pre-screening of semi-automatic fake news detection, and more importantly, bring us one step closer to solving fully automated fake news detection.

Initially, we focused on learning and utilizing useful features such as topic and sentiment information. Considering the purpose of hyper-partisan news, we believed that the stance on politically sensitive topics would be crucial in determining hyperpartisanship. However, experiments showed that the dataset contains some inherent noise that acted as a big barrier to learning a good classifier: 1) *noisy text* inputs from an article that contain domain-specific (i.e. political) words, slangs and spelling mistakes which are likely to be out of vocabulary (OOV). 2) *noisy labels* that mainly resulted from using publisher-level information for labeling articles (i.e. all articles from left/right-wing publishers are labeled as "hyper-partisan". For more detail, refer to Section 2).

Nevertheless, human-labeled large-scale dataset creation is a very expensive and time-consuming task, thus, it is crucial to find a better way to utilize this weakly-labeled dataset. Therefore, we experimented with reducing noise to help models learn better. In our work, we apply a semi-supervised pseudo-labeling to de-noise the dataset (Figure 1) and leverage the state-of-the-art pre-trained BERT (Devlin et al., 2018) to obtain a better representation of the noisy input.

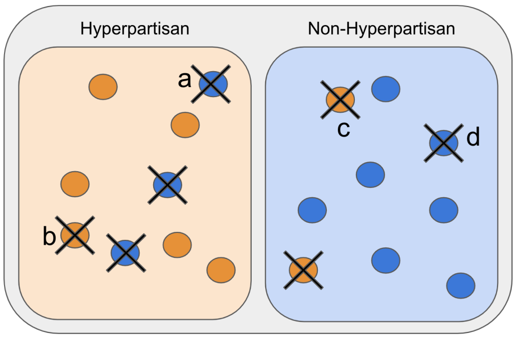


Figure 1: Illustration of filtering. Sample *a* and *c* are removed as pseudo-label ≠ by-publisher label. Sample *b* and *d* are removed as their prediction confidence was below threshold.

[*] These two authors contributed equally.

2 Data Analysis

Label Items	Train Set %	Val Set %
right	25%	25%
right-center	7.1%	8.8%
left	25%	25%
left-center	11.7%	15.7%
least bias	**31.2%**	**25.5%**
all	100%	100%
hyperpartisan	50%	50%
mainstream	50%	50%

Table 1: Data statistic of hyperpartisan and political orientation on by-publisher dataset.

We use a publicly available dataset "SemEval 2019 Task 4 - Hyperpartisan News Detection" [1] that are labeled in two different ways - publisher level and article level.

- Publisher-level (by-publisher): A total of 750K articles are labeled based on the political orientation of the publisher, without considering the content. It has an equal ratio (375K/375K) between hyperpartisan and non-hyperpartisan. Among the hyperpartisan samples, there's an equal ratio (187.5K/187.5K) between right and left political orientation.

- Article-level (by-article): A total of 645 articles labeled on article-level by checking the actual content. The data contains only articles for which a consensus among the crowd-sourcing workers existed. Of these, 238 (37%) are hyperpartisan and 407 (63%) are not.

2.1 Discussion on the Inherent Noise

By using human judgment, we discovered that some article samples did not always have the correct labels. Since the political orientation of the publisher was used as a sole criterion for the labels, such labeling noise is not surprising. It cannot be guaranteed that all articles from a hyper-partisan publisher are hyper-partisan. Another possible reason for such noise could be from not having enough non-hyper-partisan publishers (i.e. The percentage of "least bias" label items in Table 1 is not 50%), thus, treating news from "right-center" and "left-center" publishers also as non-hyper-partisan.

[1]https://zenodo.org/record/1489920#.XAAoMJMzYWq

3 Methodology

In this section, we describe how we did de-noising in our system in Figure 2. Our system consists of two steps: 1) Obtaining de-noised by-publisher dataset by leveraging clean by-article dataset. 2) Leveraging the de-noised by-publisher dataset and pre-trained BERT to train our final model. Note that our code is publicly available for reproducibility [2].

3.1 Step 1: Filter Noise by Leveraging Pseudo-labeling

To deal with the noise in the labels, we utilize pseudo-label for filtering out noisy labels from data-level (Figure 1). Pseudo-labeling is one of the semi-supervised learning methods, which approximates the labels of unlabeled data by using a model (M) trained on the labeled dataset. Originally, pseudo-labeling directly takes the prediction from the model M as the label. This could result in the final model trained on both human-labeled and pseudo-labeled to be bounded by the accuracy of the model M.

To avoid this problem: 1) We use the original by-publisher label as the constraint. We filter out data points that have a mismatch in the by-publisher label and pseudo-label to obtain cleaner by-publisher. 2) To be robust to the errors made by the model M, we set some thresholds to only use pseudo-labels with relatively high confidence. We only consider prediction scores that is bigger/smaller by $margin = 0.2$ than the mid-value (0.5). By doing so, we can filter out noisy labels with the guarantee that the noise level would be at worst kept the same; the size of our de-noised dataset is approximately 32K for both labels, which is 8.5% of original data. Note that in our system, the model M is a binary classifier trained on top of fine-tuned BERT (refer to step 2) using clean by-article dataset.

3.2 Step 2: Obtain Better Input Representation using BERT

The article texts are noisy with a lot of political words, slangs, and even spelling mistakes, many of which are out of vocabulary (OOV) and harmful to the sentence-level and article-level representation learning. We leverage state-of-the-art pre-trained language representation model BERT to

[2]https://github.com/zliucr/hyperpartisan-news-detection

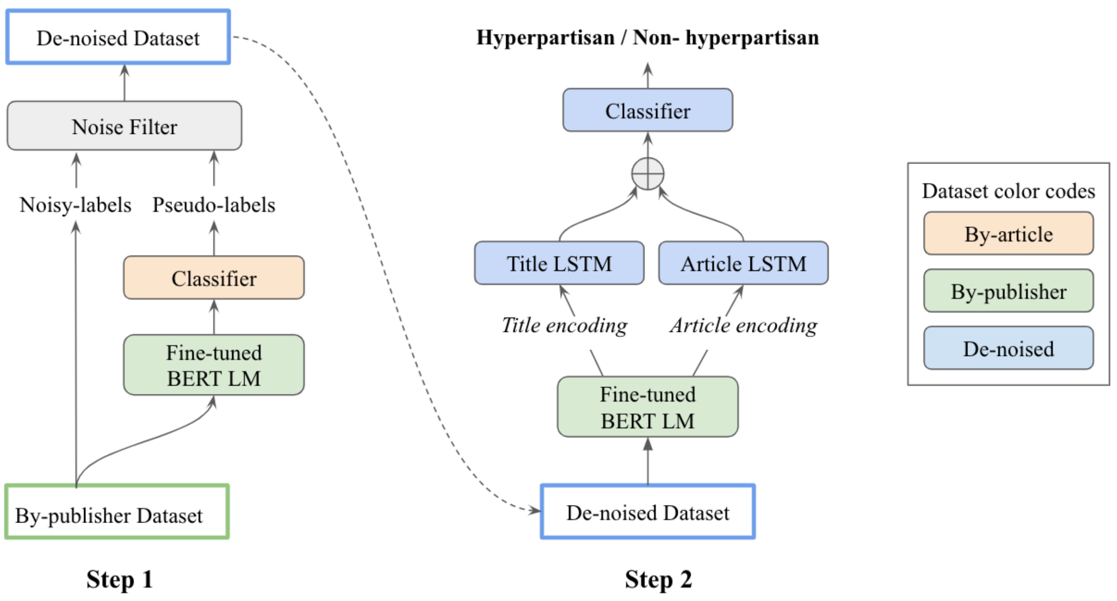


Figure 2: Architecture of the proposed system. Colors represent the dataset used to train the corresponding model.

eliminate OOV problem, since it uses byte-pairs vocabulary, and for a better input representation.

Since the pre-trained BERT model is trained on BooksCorpus and Wikipedia which are not directly relevant to news, we fine-tune the BERT, as in original paper, using our by-publisher news dataset to learn a better representation for our data domain. We build our proposed model by adding title LSTM and article LSTM on top of the fine-tuned BERT model to extract features that are concatenated and fed into the final binary classifier. We train our final classifier using the filtered dataset from Step 1.

4 Experiments and Analysis

4.1 Experimental Setup

We use BERT$_{BASE}$ model from (Devlin et al., 2018) which has 12 layers (i.e., Transformer blocks) with a hidden size of 768 and 12 self-attention heads. In step 1, the parameters of BERT model were fixed after fine-tuned on by-publisher datset, then we trained classifier on by-article dataset by using 16 batch size. We used 10-fold cross-validation to choose the parameters of the classifier, since the size of by-article dataset is small. In step 2, we used 16 batch size to train our LSTM for article model with a hidden size of 300 and LSTM for title model with a hidden size of 100. The classifiers in step 1 and 2 both consist of two linear layers with ReLU and batch normalization in between.

For the evaluation metric, we mainly considered accuracy and F1 score as the main indicator of performance. For analysis purpose, we also report precision and recall. In the competition, there were two types of test sets (i.e. by-publisher test set and by-article test set). However, all of the reported results are obtained from the by-article test set for fair and correct comparison.

4.2 Results

We ran the experiment on 3 baseline models for comparison and simple ablation study of our approach, and the results are presented in Table 2.

- **2 LSTM + Attention + Fine-tuned Classifier** ($LSTM_{ft}$)
  A baseline model consisting of 2 LSTM models (one for the title, and another for the article) with attention layers and a multi-layer perceptron (MLP) as a classifier on the top. It was trained on by-publisher dataset directly, then fine-tuned using the by-article dataset.

- **Pre-trained BERT+Classifier** ($BERT_{pt}$)
  This model uses the original pre-trained BERT model to encode both article and title, which get fed into multilayer perceptron (MLP) to predict the hyper-partisanship of the given article. The parameters of the BERT model was fixed when training the MLP classifier on the by-article data.

Models	Accuracy	Precision	Recall	F1
$LSTM_{ft}$	0.6258	0.5838	**0.8758**	0.7006
$BERT_{pt}$	0.5669	**0.8621**	0.1592	0.2688
$BERT_{ft}$	0.6592	0.8378	0.3949	0.5368
$BERT_{ft}$ + De-noise	**0.758**	0.744	0.7866	**0.7647**

Table 2: Results of our model and other baseline models on the final by-article test set.

- **Fine-tuned BERT+Classifier** ($BERT_{ft}$)
  For this model, everything is kept the same as $BERT_{pt}$ except for the fact that pre-trained BERT was fine-tuned using by-publisher dataset.

Firstly, we can observe that simply using pre-trained BERT ($BERT_{pt}$) to represent input cannot out perform LSTM model entirely trained on hyperpartisan dataset. However, by fine-tuning BERT using our dataset ($BERT_{ft}$), we gain improvement in performance by approximately 10% in accuracy, outperforming $LSTM_{ft}$ by $\approx 3\%$. Hence, we can infer that by injecting some domain-specific data into the original BERT, we can obtain an improved text representation for solving our task. Note that the model sizes for Pre-trained BERT + Classifier and Fine-tuned BERT + Classifier are the same.

Secondly, by training the same fine-tuned BERT model on the de-noised dataset mentioned in Section 3.1, we observed a big improvement in accuracy, F1 and recall by $\approx 10\%$, $\approx 23\%$ and $\approx 40\%$ respectively. This clearly illustrates the power of de-noising the dataset using pseudo-labels as auxiliary reference label. We also would like to emphasize that we did not use any ensemble learning or tricks, which normally gives extra $1-2\%$ gain in the final performance. Our system ranked 11 out of 43 teams that participated.

Lastly, we would mention that our $LSTM_{ft}$ model is a strong baseline because it was able to achieve a high score in the by-publisher test set by obtaining 0.663 and 0.694 for accuracy and F1 respectively (rank 5/28).

4.3 Interesting Analysis

Although our current system does not make direct use of topic information, we present an interesting result obtained while experimenting with topic modeling for hyper-partisanship detection. We used Latent Dirichlet allocation (LDA) for topic modeling, and the results empirically showed interesting relationships between topics and hyper-partisanship. Sensitive topics such as war and political parties tend to have more hyperpartisan news than neutral-topics such as school and sports games. We believe that leveraging such information would be helpful in future works.

5 Related Works

In this part, we briefly review the prior work in language representation as well as the semi-supervised learning method we used.

5.1 Language Representation

(Kiros et al., 2015) tried to learn sentence embedding by reconstructing the surrounding sentences of an encoded passage. (Peters et al., 2018) proposed to extract context-sensitive features from a language model. (Devlin et al., 2018) jointly conditioned on both left and right context and obtained state-of-the-art results on eleven natural language processing tasks.

5.2 Semi-supervised Learning

(Triguero et al., 2015) provided a survey of self-labeled methods for semi-supervised classification. (Zhu and Goldberg, 2009) showed self-labeled techniques are typically divided into self-training and co-training. (Lin et al., 2018) proposed semi-supervised learning to leverage a small amount of user-comment data to train a model and then expand the dataset by that trained model.

6 Conclusion

To conclude, we successfully removed noise from data-level and model-level by utilizing pseudo-labels and state-of-the-art BERT. Compared to other baselines, our de-noised model managed to outperform all, and achieve rank 11 from 42 teams. Since the cost of manual labeling fake news data is expensive, our approach to obtain cleaner and larger dataset by leveraging smaller but clean dataset is meaningful.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Benjamin D Horne and Sibel Adali. 2017. This just in: fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. In *Eleventh International AAAI Conference on Web and Social Media*.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Nayeon Lee, Chien-Sheng Wu, and Pascale Fung. 2018. Improving large-scale fact-checking using decomposable attention models and lexical tagging. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1133–1138.

Zhaojiang Lin, Genta Indra Winata, and Pascale Fung. 2018. Learning comment generation by leveraging user-generated data. *arXiv preprint arXiv:1810.12264*.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2017. Automatic detection of fake news. *arXiv preprint arXiv:1708.07104*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 231–240. Association for Computational Linguistics.

Baoxu Shi and Tim Weninger. 2016. Fact checking in heterogeneous information networks. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 101–102. International World Wide Web Conferences Steering Committee.

Isaac Triguero, Salvador García, and Francisco Herrera. 2015. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information systems*, 42(2):245–284.

William Yang Wang. 2017. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*.

You Wu, Pankaj K Agarwal, Chengkai Li, Jun Yang, and Cong Yu. 2014. Toward computational fact-checking. *Proceedings of the VLDB Endowment*, 7(7):589–600.

Xiaojin Zhu and Andrew B Goldberg. 2009. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130.

The Sally Smedley Hyperpartisan News Detector at SemEval-2019 Task 4: Learning Classifiers with Feature Combinations and Ensembling

Kazuaki Hanawa*[1,2], Shota Sasaki*[1,2], Hiroki Ouchi[1,2], Jun Suzuki[2,1], Kentaro Inui[2,1]
(* equal contribution)
[1]RIKEN AIP, [2]Tohoku University
{hanawa, sasaki.shota, hiroki.ouchi, jun.suzuki, inui}
@ecei.tohoku.ac.jp

Abstract

This paper describes our system submitted to the formal run of SemEval-2019 Task 4: Hyperpartisan news detection. Our system is based on a linear classifier using several features, i.e., 1) embedding features based on the pre-trained BERT embeddings, 2) article length features, and 3) embedding features of informative phrases extracted from the by-publisher dataset. Our system achieved 80.9% accuracy on the test set for the formal run and got the 3rd place out of 42 teams.

1 Introduction

Hyperpartisan news detection (Kiesel et al., 2019; Potthast et al., 2018) is a binary classification task, in which given a news article text, systems have to decide whether or not it follows a hyperpartisan argumentation, i.e., "whether it exhibits blind, prejudiced, or unreasoning allegiance to one party, faction, cause, or person" (2019). As resources for building such a system, the by-publisher and by-article datasets are provided by the organizer. The by-publisher dataset is a collection of news articles labeled with the overall bias of the publisher as provided by BuzzFeed journalists or MediaBiasFactCheck.com. The by-article dataset is a collection labeled through crowdsourcing on an article basis. This data contains only the articles whose labels are agreed by all the crowd-workers. The performance measure is accuracy on a balanced set of articles.

Our system is based on a linear classifier using several types of features mainly consisting of 1) embedding features based on the pre-trained BERT embeddings (Devlin et al., 2018) and 2) article length features and 3) embedding features of informative phrases extracted from by-publisher dataset. Our system achieved 80.9% accuracy on the test set for the formal run and got 3rd place out of 42 teams in the formal run.

2 System Description

This section first presents an overview of our system and then elaborate on the feature set.

2.1 Overview of Our System

Our system is based on a linear classifier that models the conditional probability distribution over the two labels (positive or negative) given features. Let $\mathbf{f}$ be a feature vector. $\mathbf{W}$ denotes a trainable weight matrix, and $\mathbf{b}$ is a trainable bias vector, where $\mathbf{f} \in \mathbb{R}^D$, $\mathbf{W} \in \mathbb{R}^{D\times 2}$ and $\mathbf{b} \in \mathbb{R}^2$, respectively. Then, we compute the conditional probability as follows:

$$\mathbf{y} = \mathtt{softmax}(\mathbf{W}^\top \mathbf{f} + \mathbf{b}). \tag{1}$$

where, $\mathtt{softmax}(\cdot)$ represent the softmax function that receives an N-dimensional vector $\mathbf{x}$ and returns another N dimensional vector, namely:

$$\mathtt{softmax}(\mathbf{x}) = \frac{\exp(\mathbf{x})}{\sum_i \exp(x_i)}, \tag{2}$$

and $\mathbf{x} = (x_1, \ldots, x_N)^\top$. After the softmax computation, we obtain the two-dimensional vector $\mathbf{y} \in \mathbb{R}^2$. We assume that the first dimension of this vector represents the probability of the positive label, and the second one represents that of the negative label.

To boost the performance, we concatenate three types of features, $\mathbf{f}_1$, $\mathbf{f}_2$, and $\mathbf{f}_3$, into the single feature vector $\mathbf{f}$, where $\mathbf{f}_1 \in \mathbb{R}^{D_1}$, $\mathbf{f}_2 \in \mathbb{R}^{D_2}$ and $\mathbf{f}_3 \in \mathbb{R}^{D_3}$ and $D = D_1 + D_2 + D_3$.

As $\mathbf{f}_1$, $\mathbf{f}_2$ and $\mathbf{f}_3$, we design the following features.

- $\mathbf{f}_1$: BERT feature (Section 2.2)

- $\mathbf{f}_2$: Article length feature (Section 2.3)
- $\mathbf{f}_3$: Informative phrase feature (Section 2.4)

For training our classifiers, we used only the `by-article` dataset but not the `by-publisher` dataset. This is because the labels of the `by-publisher` dataset turned out rather noisy. In our preliminary experiments, we found that the performance drops when training the classifiers on the `by-publisher` dataset.

Furthermore, we apply the following three techniques.

1. **Word dropout**: We adopted word dropout (Iyyer et al., 2015) for regularization. The dropout rate was set to 0.3.
2. **Over sampling**: As mentioned above, the gold label distribution of the training set is unbalanced while that of the test set is balanced. We deal with this imbalance problem by sampling 169 (407 − 238) extra examples from hyperpartisan data.
3. **Ensemble**: We trained 100 models with different random seeds. In addition, we trained models for 40, 50, 60 and 70 epochs for each seed. Consequently, we finally average the output of these 400 models.

2.2 BERT Feature

Our system uses BERT (Devlin et al., 2018). As a strategy for applying BERT to downstream tasks, Devlin et al. (2018) recommends a fine-tuning approach, which fine-tunes the parameters of BERT on a target task. In contrast, we adopt a feature-based approach, which uses the hidden states of the pre-trained BERT in a task-specific model as input representation. A main advantage of this approach is computational efficiency. We do not have to update any parameters of BERT. Once we calculate a fixed feature vector for an article, we can reuse it across all the models for ensemble.

In our system, we used BERT to compute a feature vector $\mathbf{f}_1$ for an input article. Specifically, we first fed an article to the pre-trained BERT model as input and extracted the representations of all the words from the top four hidden layers. Then, to summarize these representations into a single feature vector $\mathbf{f}_1$, we tried the following three methods.

1. **Average**: Averaging the representations of all the words in an article.
2. **BiLSTM**: Using the representations as input to BiLSTM. This is the same method as the best performing one reported by Devlin et al. (2018).
3. **CNN**: Using the representations as input to CNN in the same way as Kim (2014).

As we describe in Section 3.2, we finally adopted the averaged BERT vectors as $\mathbf{f}_1$.

2.3 Article Length Feature

As $\mathbf{f}_2$, we design a feature vector representing the length of an input article. In our preliminary experiments, we found a length bias in hyperpartisan articles and non-hyperpartisan articles. Thus a vector representing the length bias is expected to be useful for discriminating these two types of articles.

Specifically, we define a one hot feature vector $\mathbf{f}_2$ representing distribution of the lengths of articles (the number of words in an article). To represent the length of an article as a vector, we make use of histogram bins. Consider the 100-ranged histogram bins. The first bin represents the length 1 to 100, and the second one represents the length 101 to 200. If the length of an article is 255, the value of the third bin (201 to 300) takes 1. In the same way, the third element of the length vector takes 1 and the others 0, i.e., $\mathbf{f}_2 = [0, 0, 1, 0, 0, \cdots]$. In our system, we set the dimension of the vector as $D_2 = 11$, whose last (11-th) element corresponds to the length longer than 1000.

2.4 Informative Phrase Feature

In the development set, we found some phrases informative and useful for discriminating whether or not a given article is hyperpartisan. We extracted such informative phrases and mapped them to a feature vector $\mathbf{f}_3$. In this section, we first explain the procedure of extracting informative phrases, and then we describe how to map them to a feature vector.

2.4.1 Phrase Set Creation

To create an informative phrase set, we exploit the `by-publisher` articles. Basically, we take advantage of chi-squared statistics of N-grams ($N = 1, 2, 3$).

Creation of S_h First, we calculate each chi-squared value χ_{x_i} of N-gram x_i appearing in the by-publisher articles as follows:

$$\chi_{x_i} = \frac{(O - E)^2}{E}. \quad (3)$$

O and E are defined as follows:

$$O = f_{\text{false}}(x_i), \quad (4)$$

$$E = \frac{T_{\text{false}} \times f_{\text{true}}(x_i)}{T_{\text{true}}}, \quad (5)$$

where $f_{\text{true}}(\cdot)$ and $f_{\text{false}}(\cdot)$ are functions that calculate the frequency of x_i in hyperpartisan articles and non-hyperpartisan articles, respectively. T_{true} and T_{false} are the summation of the frequency of all N-grams in hyperpartisan articles and non-hyperpartisan articles, respectively.

Then, based on the chi-squared values χ_{x_i}, we sort and select top-M N-grams. We can obtain a typical N-gram set (hereinafter, referred to as S_h) that is informative for judging whether an article is hyperpartisan or not.[1] In our system, we use $M = 200,000$.

S_h can mostly catch the characteristics of hyperpartisan articles. However, S_h may include some N-grams that are typical of a certain publisher. This is because the by-publisher dataset is labeled by the overall bias of the publisher as provided by BuzzFeed journalists or MediaBiasFactCheck.com.

Creation of S_p To remedy this problem, we create another phrase set S_p consisting of N-grams that are typical of a publisher, and exclude them from S_h.

Here we consider a certain publisher p_l. First, we calculate each chi-squared value of N-gram x_i in the same way as Eq 3,4,5 for S_h, but here we consider $true$ and $false$ as appearing in articles of publisher p_l and articles of other publishers, respectively, instead of appearing in hyperpartisan articles and non-hyperpartisan articles. Then, we pick up only N-gram x_i where $f_{\text{true}}(x_i)$ is less than $f_{\text{false}}(x_i)$ and sort all of them by the chi-squared values. This is because we aim to exclude only N-grams that are typical of a certain publisher.

Next, we try four types of ways to create S_{p_l} from sorted χ_x value statistics. Here j denotes the index of the N-gram list sorted by χ_x values, i.e., χ_{x_1} is the highest value in all calculated χ_x values.

[1] Actually, we cut off N-gram x if $f_{\text{true}}(x)$ is more than 100,000 for S_h.

1. **Top-T_o**: The first setting is to select top-T_o N-grams. Concretely, S_{p_l} is defined as follows:
$$S_{p_l} = \{x_j | j \leq T_o\}. \quad (6)$$
2. **χ-based**: The second setting is to select N-grams based on χ values. Concretely, S_{p_l} is defined as follows:
$$S_{p_l} = \{x_j | \chi_{x_j} > T_c\}. \quad (7)$$
3. **f_{true}-based**: The third setting is to select N-grams based on $f_{\text{true}}(x_j)$ values. Concretely, S_{p_l} is defined as follows:
$$S_{p_l} = \{x_j | f_{\text{true}}(x_j) > T_f, j \leq T_m\}. \quad (8)$$
4. **f_{true}-f_{false} ratio-based**: The fourth setting is to select N-grams based on ratios between f_{true} and f_{false}. Concretely, S_{p_l} is defined as follows:
$$S_{p_l} = \left\{ x_j \,\middle|\, \frac{f_{\text{true}}(x_j)}{f_{\text{false}}(x_j)} > T_r, j \leq T_m \right\}. \quad (9)$$

T_o, T_c, T_f, T_r and T_m are hyper-parameters[2].

Next, we obtain S_p defined as follows:

$$S_p = \bigcup_l S_{p_l}. \quad (10)$$

At last, we obtain an filtered N-gram set S defined as follows:

$$S = S_h \setminus S_p. \quad (11)$$

2.4.2 Phrase Embedding

We map each of the obtained N-gram phrase set S to a feature vector $\mathbf{f}_3$. We exploited GloVe vectors (Pennington et al., 2014) instead of one-hot vectors in order to facilitate generalization.

First, we enumerate N-grams included in an article and compute each N-gram vector. Each vector is the average of GloVe vectors of included words. For example, the vector for the phrase "news article" is computed as follows:

$$\frac{\texttt{GloVe}(\text{news}) + \texttt{GloVe}(\text{article})}{2}.$$

Here, $\texttt{GloVe}(w)$ denotes the GloVe (glove.840B[3]) vector of the word w. Then, we compute $\mathbf{f}_3$ as the average of all N-gram vectors included in the article.

[2] In our experiments, we fix T_m to 200,000.

[3] https://nlp.stanford.edu/projects/glove/

Method	Accuracy
Average	0.760
BiLSTM	0.712
CNN	0.758

Table 1: Accuracy of hyperpartisan classification for each operation on BERT vectors.

3 Experiments

3.1 Settings

We trained linear classifiers on the `by-article` dataset (not on the `by-publisher` dataset). In order to estimate the performance in the test set with each setting, we conducted 5-fold cross validation on `by-article` dataset. For optimization of the classifiers, we use Adam with learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$. We set the minibatch size to 32. Note that we did not take ensemble approach in the experiments we report in Section 3.2 and Section 3.3 for efficiency.

3.2 Operation on BERT Vectors

We conducted experiments on each of the three methods for BERT vectors (`BERT-Base, Uncased`[4]) mentioned in Section 2.2. In this experiment, we only used $\mathbf{f}_1$ as a feature vector $\mathbf{f}$, i.e., without using $\mathbf{f}_2$ and $\mathbf{f}_3$.

Table 1 shows the performance in each setting. The averaging method was the best performance this time. We therefore decided to adopt average BERT vectors as $\mathbf{f}_1$ for the evaluation of the formal run. In addition, we also used averaged BERT vectors as $\mathbf{f_1}$ in the following experiments.

3.3 Method to Create N-gram Set

As mentioned in Section 2.4, we examined which method is the best to create an informative N-gram set S (and $\mathbf{f}_3$ derived from them). In this experiment, we also used $\mathbf{f}_1$ and $\mathbf{f}_2$ with $\mathbf{f}_3$ as a feature.

Table 2 shows the performance in each setting. The performance was the best when we adopted Top-T_o ($T_o = 100$) for S_p creation. We therefore used $\mathbf{f}_3$ created in this setting.

3.4 Ablation

To verify the contribution of each three types of features, we conducted feature ablation experiments. In addition, we investigated to what extent ensemble approach improve the performance. In this experiments, we use only 10 (not 100) different random seeds for ensemble due to time constraints.

[4]https://github.com/google-research/bert

Method	Accuracy
Top-T_o ($T_o = 100$)	0.777
Top-T_o ($T_o = 1000$)	0.771
χ-based ($T_c = 20000$)	0.752
f_{true}-based ($T_f = 50$)	0.754
f_{true}-based ($T_f = 150$)	0.764
f_{true}-f_{false} ratio-based ($T_r = 0.5$)	0.754
f_{true}-f_{false} ratio-based ($T_r = 0.8$)	0.771
f_{true}-f_{false} ratio-based ($T_r = 1.0$)	0.756

Table 2: Accuracy of hyperpartisan classification for each method to create N-gram set.

Features	Ensemble	Accuracy
$\mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}$	true	0.788
$\mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}$	false	0.777
$\mathbf{f_1}, \mathbf{f_2}$	false	0.769
$\mathbf{f_1}$	false	0.760

Table 3: Result of ablation.

Table 3 shows the performance in each setting. We found that $\mathbf{f_2}$ and $\mathbf{f_3}$ improved the accuracy by about 0.01, respectively. Additionally, by using the ensemble method, the accuracy increased by about 0.01.

4 Conclusion

We described our system submitted to the formal run of SemEval-2019 Task 4: Hyperpartisan news detection. We trained a linear classifier using several features mainly consisting of 1) BERT embedding features, 2) article length features indicating the distribution of lengths of articles and 3) embedding features derived from filtered N-grams that are typically found in hyperpartisan articles. Our system achieved 80.9% accuracy on the test set for the formal run and got 3rd place out of 42 teams.

Acknowledgments

This work was supported by JST CREST Grant Number JPMJCR1301, Japan.

References

2019. Pan @ SemEval 2019 - Hyperpartisan News Detection. https://pan.webis.de/semeval19/semeval19-web/index.html.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 1681–1691.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. Semeval-2019 task 4: Hyperpartisan news detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A stylometric inquiry into hyperpartisan and fake news. In *Proceedings of 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 231–240.

Tintin at SemEval-2019 Task 4: Detecting Hyperpartisan News Article with only Simple Tokens

Yves Bestgen
Centre for English Corpus Linguistics
Université catholique de Louvain
Place Cardinal Mercier, 10 1348 Louvain-la-Neuve, Belgium
yves.bestgen@uclouvain.be

Abstract

Tintin, the system proposed by the CECL for the Hyperpartisan News Detection task of SemEval 2019, is exclusively based on the tokens that make up the documents and a standard supervised learning procedure. It obtained very contrasting results: poor on the main task, but much more effective at distinguishing documents published by hyperpartisan media outlets from unbiased ones, as it ranked first. An analysis of the most important features highlighted the positive aspects, but also some potential limitations of the approach.

1 Introduction

This report presents the participation of Tintin (Centre for English Corpus Linguistics) in Task 4 of SemEval 2019 entitled Hyperpartisan News Detection. This task is defined as follows by the organizers[1]: "Given a news article text, decide whether it follows a hyperpartisan argumentation, i.e., whether it exhibits blind, prejudiced, or unreasoning allegiance to one party, faction, cause, or person."

This question is related to the detection of fake news, a hot topic in our internet and social media world (Pérez-Rosas et al., 2017). There are, however, essential differences between these two tasks. An article can be hyperpartisan without mentioning any fake content. Another difference is that it is a news article (or even a claim) that is fake whereas a news article but also a media outlet (or publisher) can be considered as hyperpartisan. The challenge organizers took these two possibilities (i.e. an article or a publisher can be hyperpartisan) into account by offering two test sets. The main test set, the labels-by-article one, contained documents that had been assessed as hyperpartisan or not by human judges, while the documents in the secondary test set, the labels-by-publisher one, had been categorized according to whether their publishers were considered to be hyperpartisan or not by organizations that disseminate this type of evaluation. In both these test sets, participants had to decide whether a document expresses a hyperpartisan point-of-view or not.

If the main task is particularly interesting, the secondary task is also relevant because it is about achieving through an automatic procedure what a series of organizations manually perform in a way that is sometimes called into question as to its impartiality and quality (Wilner, 2018). However, in this context, the task would preferably be evaluated, not at the document level, but at the publisher level by providing several documents from a publisher and asking whether the publisher is biased or not. Nevertheless, it can be assumed that many systems developed for categorizing publishers will start by evaluating each document separately and thus getting good performance in the current secondary task is at least a first step.

To take up these tasks, the question is how to determine automatically whether a document is hyperpartisan or not. This question has not attracted much attention in the literature, but, very recently, Potthast et al. (2018) proposed to use stylometric features such as characters, stop words and POS-tag n-grams, and readability measures. They compared the effectiveness of this approach to several baselines including a classical bag-of-words feature approach[2] (Burfoot and Baldwin, 2009). Their stylistic approach obtained an accuracy of 0.75 in 3-fold cross-validation in which publishers present in the validation fold were unseen during the learning phase. The bag-of-words feature

[1]https://pan.webis.de/semeval19/semeval19-web

[2]More specifically, Potthast et al. (2018) used the frequency, normalized by the document length, of the tokens of at least two characters that occurred in at least 2.5% of the documents in the collection.

approach obtained an accuracy of 0.71, which is not much lower. These results were obtained on a small size corpus (due to the cost of the manual fact-checking needed for the fake-news part of the study) containing only nine different publishers. It is therefore not evident that this corpus was large enough to evaluate the degree of generalizability of the bag-of-words approach, especially since Potthast et al. (2018, p. 233) emphasizes that using bag-of-words features potentially related to the topic of the documents renders the resulting classifier not generalizable. In contrast, the datasets prepared for the present challenge are significantly larger since the latest versions available contain more than 750,000 documents and more than 240 different media outlets.

Therefore, it seemed interesting to evaluate the effectiveness of a bag-of-words approach for the labels-by-publisher task, the one used by Potthast et al. (2018). This is the purpose of this study. Another reason why I chose to focus on the labels-by-publisher task is that I was unclear about what could be learned on the basis of the labels-by-publisher sets for the labels-by-article test set. If one can think that some publishers almost always distribute hyperpartisan articles, it seems doubtful that this is the case for all of them.

The next sections of this paper describe the datasets, the developed system, and the obtained results as well as an analysis of the most important features.

2 Data

As explained in Kiesel et al. (2019), several datasets of very different sizes were available for this challenge. The learning labels-by-publisher set contained 600,000 documents form 158 media outlets in its final version. The corresponding validation set contained 150,000 documents from 83 media outlets, and the test set consisted of 4,000 documents. The first labels-by-article set provided to the participants contained 645 documents and was intended for fine-tuning systems developed on the labels-by-publisher sets. The test set contained 628 documents.

Some of these datasets could be downloaded while those used to perform the final test were hidden on a TIRA server (Potthast et al., 2019). An important feature of these data is that no publisher in a dataset is present in any other dataset. This has the effect of penalizing (usefully) any system that learns to categorize on the basis of the publishers since generalization to unseen media outlets should be problematic.

3 System

3.1 The Bag-of-Words Feature Approach

The developed system, which implements the bag-of-words approach, is very classical. It includes steps for preprocessing the data, reading the documents, creating a dictionary of tokens (only unigram tokens as bigrams did not appear to improve performance), and producing the file for the supervised learning procedure. It was written in C, with an initial data cleaning step in Perl, and was thus very easy to install on a TIRA server. In this section, only a few implementation details are mentioned.

During preprocessing, a series of character sequences like *;amp;amp;amp;*, *&amp;#160;* and *&amp;amp;lt;* were regularized. When reading a document (both the title and the text), strings were split by separating the following characters when they were at the beginning or end of the strings and they were outputted separately: ’ * ” ? . ; : / ! ,) (} { [ ] -. Alphabetic characters were lowercased. A binary feature weighting scheme was used.

3.2 Supervised Learning Procedure

During the development and test phases of the challenge, the models were build using two solvers available in the LIBLINEAR package (Fan et al., 2008), the L2-regularized L2-loss support vector classification (-s 1) and the L2-regularized logistic regression (-s 7), which resulted in equivalent performance. The regularization parameter C was optimized on the labels-by-publisher validation set using a grid search.

4 Analyses and Results

4.1 Official Results

On the main task of the challenge, the Tintin system obtained an accuracy of 0.656, ranking 27th out of 42 teams, very far from the best teams who scored 0.82.

Twenty-nine teams submitted a system for the labels-by-publisher task. Tintin ranked first, with an accuracy of 0.706. This level of performance is identical to that obtained by Potthast et al. (2018) bag-of-words model in their experiments on a significantly smaller dataset.

In general, the performances of the different teams on the second task were much lower than on the main task. Tintin, on the other hand, achieved a better score on the second task. It is not the only system in this case since, of the 28 teams that participated in the two tasks, three others also scored better in the second task and one team only participated in this task. Reading the papers describing these systems will make it possible to know if these teams have also chosen to favor the secondary task. It is also noteworthy that the difference between the two best teams is much greater in the secondary task (0.706 vs. 0.681) than in the main task (0.822 vs. 0.820).

4.2 Analysis of the Most Important Features

In order to get an idea of the kind of features underlying the system's efficiency in the secondary task, the 200 features (and thus tokens) that received the highest weights (in absolute value) in the logistic regression model[3] were examined.

Table 1 shows the ten features that received the highest weights as well as a series of features selected because of their interest to understand how the system works. Positive weights indicate that the feature predicts the hyperpartisan category, while negative weights are attributed to features that are typical of the non-biased category. The table gives in addition to the token and the weight, the number of publishers (#Pub) and the number of documents (#Doc) in which that token appears for each of the two categories to be predicted. The maximum percentage of documents a publisher represents in each category is also provided (Max%). The percentage for the category that this feature predicts is boldfaced.

As expected, some of the most important features are typical of a single publisher like *globalpost*, which is present in 750 times more non-biased than hyperpartisan documents, but 99.76% of the non-biased documents come from the same publisher (*pri.org*). Other tokens are not so strongly associated with a single publisher. In the 8th position, the token *h/t*, a way of acknowledging a source, is present in 53 hyperpartisan media outlets and 63% of the documents of this category in which it occurs are not found in the publisher that contains the most (*dailywire.com*). *Jan* is an even more obvious example of features that are not tied to a single publisher.

There are also in these particularly important features some tokens that might not be seen as unexpected such as *leftists(s)*, *shit*, *beast*, *right-wing*, *hell...* Other features, such as *fla*, *beacon*, *alternet* or *via*, are not related to a single publisher, but their usefulness for categorizing unseen media outlets is no less debatable. For instance, *via* can be used in many different contexts such as *via twitter*, *transmitted to humans via fleas*, *linking Damascus to Latakia and Aleppo via Homs*. It is therefore widespread. However, its usefulness in categorizing *unseen* media outlets is not necessarily obvious since some part of its weight results from its occurrence in all of the 976 documents from *thenewcivilrightsmov* as each of these documents offers to *subscribe to the New Civil Rights Movement via email*.

These observations lead to wonder whether the system does not show a strong variability of efficiency according to the unseen publishers to predict, working well for some, but badly for others. It was not possible to evaluate this conjecture by analyzing the system accuracy for the different publishers in the test set since it is not publicly available. However, an indirect argument in its favor is provided by the meta-learning analyses done by the task's organizers that suggest that some publishers are much easier to predict than others. For these analyses, each set was randomly split into two samples (2668 vs. 1332 for the labels-by-publisher test set) and submitted to a majority voting procedure. As this procedure is unsupervised, the expected value of the difference in accuracy between the two samples is 0. This was not the case for the labels-by-publisher task since it was larger than 0.23, an extremely significant difference (Chi-square test). The most obvious explanation is to consider that the need to put each publisher in only one sample leads to a non-random distribution in which the publishers of one sample are much easier to predict.

5 Conclusion

The Tintin system, developed for the Hyperpartisan News Detection task, is extremely simple since it is exclusively based on the document tokens. If its performance on the main task was poor, it ranked first when it was used to discriminate documents published by hyperpartisan media outlets from unbiased ones. An analysis of the

[3]As the features are binary coded, weight is the sole factor that affect the classification function for an instance.

			Unbiased			Hyperpartisan		
Rank	**Token**	**Score**	**#Pub**	**#Doc**	**Max%**	**#Pub**	**#Doc**	**Max%**
1	globalpost	-2.40	4	10506	**99.7**	10	14	21.4
2	n.m	-1.94	17	17951	**96.1**	18	107	21.4
3	upi	-1.64	15	7541	**94.6**	24	186	48.3
4	>	-1.36	13	2022	**87.2**	27	242	51.2
5	_	-1.17	8	7516	**99.4**	9	170	34.1
6	©	1.16	15	1821	76.7	25	8840	**53.1**
7	h/t	1.06	11	71	33.8	53	3016	**36.8**
8	fe	-1.03	20	13797	**97.4**	28	294	40.4
9	et	0.95	29	2326	28.3	76	13306	**84.4**
10	jan	-0.90	38	19428	**27.2**	75	4937	45.7
22	trump's	0.73	28	2966	31.6	62	9164	**39.2**
35	via	0.61	37	10738	31.3	104	24279	**18.1**
62	fla	-0.51	26	3481	**36.2**	47	797	29.2
66	leftists	0.50	16	147	27.8	74	2984	**30.6**
67	leftist	0.49	19	895	26.0	79	4904	**35.0**
76	shit	0.47	18	136	27.2	67	2167	**39.9**
82	beast	0.46	26	887	25.7	79	3151	**30.5**
95	right-wing	0.44	26	1542	23.0	88	8608	**32.7**
97	beacon	0.43	26	569	27.5	72	2048	**25.1**
143	yesterday	0.37	32	3849	18.9	102	10018	**21.4**
171	hell	0.34	31	2518	28.9	97	8382	**35.0**
192	alternet	0.33	9	15	26.6	28	795	**33.0**

Table 1: Some of the 200 most useful features for predicting hyperpartisanship.

Figure 1: Main entrance of the Musée Hergé[4].

most important features for predicting hyperpartisanship emphasizes the presence of tokens specific to certain publishers, but also of tokens that could have some degree of generalizability.

In future work, it might be interesting to use other weighting functions than the binary one such as the bi-normal separation feature scaling (Forman, 2008) that has been shown to be particularly effective for satire detection (Burfoot and Baldwin, 2009) or BM25 which has proved useful in the VarDial challenge (Bestgen, 2017). Such development, however, would only be justified if the system is stable, that is to say, if it achieves good performance for many publishers not seen during learning. Designing a weighting function that would favor the hyperpartisan distinction while simultaneously reducing the impact of the media outlets could perhaps improve this stability.

6 Namesake: Tintin

I chose this fictitious reporter as my namesake for this task because the Musée Hergé, an unusual looking building in front of which a huge fresco represents this cartoon character, is located a few tens of meters from my office in Louvain-la-Neuve. *Tintin* is also a French interjection that means *nothing* or *No way!*

Acknowledgments

The author is a Research Associate of the Fonds de la Recherche Scientifique - FNRS.

[4]©V. Pypaert, CC BY-SA 4.0, from Wikimedia.

References

Yves Bestgen. 2017. Improving the character ngram model for the DSL task with BM25 weighting and less frequently used feature sets. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 115–123, Valencia, Spain. Association for Computational Linguistics.

Clint Burfoot and Timothy Baldwin. 2009. Automatic satire detection: Are you having a laugh? In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 161–164. Association for Computational Linguistics.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

George Forman. 2008. BNS feature scaling: an improved representation over tf-idf for svm text classification. In *Proceeding of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 263–270, New York, NY, USA. ACM.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic detection of fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3391–3401. Association for Computational Linguistics.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. TIRA Integrated Research Architecture. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 231–240. Association for Computational Linguistics.

Tamar Wilner. 2018. We can probably measure media bias. but do we want to? *Columbia Journalism Review*, Retrieved 2019-02-01.

Tom Jumbo-Grumbo at SemEval-2019 Task 4: Hyperpartisan News Detection with GloVe vectors and SVM

Chia-Lun Yeh[1], **Babak Loni**[2], **and Anne Schuth**[2]
[1]TU Delft, The Netherlands
[2]De Persgroep
`c.yeh-1@student.tudelft.nl`
`{babak.loni, anne.schuth}@persgroep.net`

Abstract

In this paper, we describe our attempt to learn bias from news articles. From our experiments, it seems that although there is a correlation between publisher bias and article bias, it is challenging to learn bias directly from the publisher labels. On the other hand, using few manually-labeled samples can increase the accuracy metric from around 60% to near 80%. Our system is computationally inexpensive and uses several standard document representations in NLP to train an SVM or LR classifier. The system ranked 4th in the SemEval-2019 task. The code is released for reproducibility[1].

1 Introduction

Bias is the inclination or prejudice for or against one person or group. News articles that contain extreme bias fail to provide fair and multi-faceted views for readers and can create polarization within the society (Bernhardt et al., 2008). A system that can detect bias in news articles is thus relevant, especially in a time where an increasing number of people consume news from online sources that might not be trustful.

The SemEval-2019 task aims to detect hyperpartisan news given the text of the news article, where hyperpartisan news is defined to be an article that overtly favors a side or view. The details of the task can be found in Kiesel et al. (2019). We are provided with a dataset of two parts. The first part is labeled by the publishers (e.g. if a publisher is decided to be a hyperpartisan source, all its articles are labeled as hyperpartisan), and split into a training and validation set with no overlapping publishers (which we will refer to as training-1 and validation-1). The second part is crowd-sourced and labeled per article (which we will call training-2).

[1]`https://github.com/chialun-yeh/SemEval2019`

Due to the large number of labeled samples, we decide to use a supervised classification approach, where features are extracted from the text and used to train a classifier. Bag-of-words (BoW), TFIDF weighting, and n-grams have been shown to be strong baselines (Hu and Liu, 2004; Wang and Manning, 2012). Other features such as Part-Of-Speech (POS), counts of sentiment and bias words have also been studied (Liu, 2012; Mukherjee and Weikum, 2015). In a similar setting, Potthast et al. (2018) uses features such as n-gram of characters, readability scores, dictionary, and the ratio of quoted words to separate hyperpartisan news from the mainstream. They trained a random forest classifier and achieved an accuracy of 75%.

Kulkarni et al. (2018) build a neural network to predict the political ideology of news articles to be either left, right or center. They combine information from the headlines, the links within an article, and the content. They use a CNN (Kim, 2014) for the headlines, a Node2Vec (Grover and Leskovec, 2016) to model the links and a hierarchical attention network (HAN) (Yang et al., 2016) to extract features from the content. They compare the model with several baselines, including a BoW LR model, a fully-connected feedforward network, and networks with only the individual components. Their proposed model performs the best. However, their system is trained and evaluated on only data with publisher labels. They randomly split them into training and testing sets, with overlapping publishers.

The main contribution of the paper is two-fold. First, we analyze the problem of using the dataset labeled by publishers, concluding that it is difficult due to the noisy labels. Second, we train SVM classifiers with different representations: TFDIF, doc2vec and GloVe pre-trained vectors. The 300-dimensional GloVe vectors obtain the best cross-validation accuracy as well as the performance metrics on the official test data.

This paper is organized as follows. In section 2, we describe the data pre-processing. In section 3, we present the two systems that we devise and explain how one motivates the other. In section 4, we present the performance of the final system. We outline our main conclusions and future work in section 5.

2 Pre-processing

Since the articles are collected from online news platforms, they contain texts that are irrelevant to the news itself. We use the following three steps to clean the data:
(a) Remove online usage including links, hashtags, @-tag, and advertisements.
(b) Remove parentheses, brackets, and curly brackets that contain additional information because the usage is often specific to publishers.
(c) Remove paragraphs that might reveal publisher information. Some publishers use headers and footers of specific patterns in their articles. We try to remove them by discarding the first and last paragraphs from the article if the article has more than two paragraphs, assuming that these two paragraphs have higher probabilities of being headers and footers. This is by no means optimal since the first paragraph often contains important content if it is not a header. Some publishers also inserted short text such as "read more here" between paragraphs. To remove these irrelevant texts that can reveal publisher pattern, we remove any paragraph with less than ten words. Any article with less than ten words after the cleaning is discarded.

We consider (a) and (b) as basic data cleaning and apply them on all data. On the other hand, (c) is a more aggressive cleaning that is done only on training-1. This is because we have a comparatively large training set where we can afford filtering out information and even entire articles.

3 System Description

3.1 System 1

In the first method, we use training-1 to train our models, validation-1 to choose hyperparameters, and training-2 to test the models. As mentioned earlier, training-1 is labeled by publishers. While a biased publisher publishes more biased articles on average, it is unlikely that all of its articles are biased. Therefore, the labels are noisy, e.g., some labels are flipped. It is, however, difficult to identify the articles that have the wrong labels without manual inspection. We assume that the publisher labels are correlated with true bias labels, thus providing information to learn bias. To have an idea of to what extent this assumption holds, we investigate training-2. We select publishers of whom at least five articles are included in the dataset and whose media bias can be retrieved from Media-Bias/FactCheck[2]. This results in a total of 24 publishers. The publisher bias ratings on the website can be roughly mapped to 7 categories, extreme-left, left, left-center, center, right-center, right, and extreme-right. In Table 1, we list these publishers along with the percentage of the articles that are rated as hyperpartisan by crowd workers. The number of articles per publisher range from 5 to 24. Figure 1 shows the percentage of hyperpartisan articles in each category. We see that left-center and center publishers indeed have considerably less percentage of hyperpartisan articles. However, right-center publishers are almost as biased as right publishers. The observation can be due to the small sample size (the high percentage is caused mainly by the publisher RealClearPolitics). In general, there is a correlation between the publisher and true hyperpartisanship.

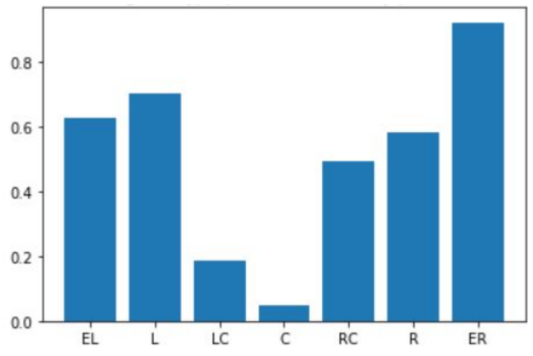


Figure 1: Percentage of hyperpartisan articles in the 7 bias categories: extreme-left (EL), left (L), left-center (LC), center (C), right-center (RC), right (R), and extreme-right (ER).

We use BoW and n-grams (n=1,2) as features, with different weighting schemes, including raw counts, binary, and TFIDF. For BoW and n-grams, the feature dimension is 50K and 500K respectively. We train two classifiers on each representation. The accuracy of the classifiers on validation-1 is listed in Table 2. We include experiments where training-1 is not cleaned with pre-

[2]https://mediabiasfactcheck.com/

processing step (c) to make sure that the step helps the task.

From the result, we observe that adding bigrams doesn't improve accuracy. We use the best model (BoW and an SVM classifier) to predict the articles in training-2. The accuracy is 56%, which is lower than the majority baseline of 63%.

Although we clean the dataset in an effort to prevent the classifier from overfitting on the publisher, it seems that the classifier cannot generalize to unseen publishers, and fails to capture bias. We also experiment with training a CNN (Kim, 2014) with the headlines, and a HAN (Yang et al., 2016) with the content. However, the two models again fail to generalize to new publishers. The observation makes us believe that the publisher labels are too noisy to be used directly to learn true bias. Another possible explanation could be that the publishers have too distinct writing styles so that the classifier focuses much on those features when learning.

3.2 System 2

Due to the observation in system 1, we decide to treat training-1 and validation-1 as unlabeled samples that can be used to train a feature extractor in an unsupervised setting. We then train the classifier using training-2. We use the first part of the data by the following two extractors.

1. TFIDF: The data is used to build vocabulary and record the inverse document frequency. All terms that occur in more than 90% of the documents are discarded, and we kept the most frequent 50K terms.

2. Doc2Vec: The data is used to train a PV-DM model proposed by Le and Mikolov (2014). We discard all terms that occur in less than 10 documents or are shorter than two characters. We train the model for 20 epochs using the implementation of gensim (Řehůřek and Sojka, 2010). When inferencing new documents, the word vectors are fixed and the model is trained for 100 epochs.

In addition, we experiment with using pre-trained word embeddings since the meaning of each word should not differ significantly in different corpora. We use vectors trained with GloVe algorithm (Pennington et al., 2014) on Wikipedia and Gigaword 5 [3]. The vectors are chosen because they are trained on Wikipedia and newswire text, which provides general knowledge and news domain specific usage. We take the vectors of each word in the document and average all the vectors. Stop words are removed, and if the document has more than 1000 words, we average over the first 1000 words (we find this to work better in our case empirically).

We also experiment with a set of features including normalized count of 5 POS tags, 6 readability scores, 8 normalized sentiment and bias word counts according to MPQA and bias lexicons (Wilson et al., 2005; Recasens et al., 2013), number of quotes, words, capitalized words, stop words, and sentences, and average length of words and sentences. This result in a total of 27 features which we call Feat.

For supervised training, we split training-2 into two sets. The first half, with 322 samples, is used to train and choose hyperparameters in a 10-fold cross validation setting. The second half, with 323 samples, is used for testing. We train LR and SVM on the features. Both linear SVM and SVM with rbf kernels are experimented with. We also have some initial experiments of single layer and two-layer neural networks of different hidden layer sizes but the small sample size makes them difficult to generalize.

4 Results

We first train LR and SVM with different GloVe vector dimensions. Table 3 shows the accuracy on the test set. SVM with rbf kernel works consistently better. The best vector dimension is 300.

We then compare different features, including TFIDF, Doc2Vec, GloVe, and the effect of adding Feat. Table 4 shows the accuracy on the test set. It shows that SVM performs better than LR, and only in the case of TFIDF does a linear SVM outperforms kernel SVM. It also shows that the pre-trained GloVe vectors achieve better performance than the vectors that are trained on our data. The ability to generalize might result from the larger corpus that is used to train the vectors. Adding simple lexical and sentiment features hurts the performance.

The three representations are furthered evaluated on another test set (the official test set of the

[3]https://catalog.ldc.upenn.edu/LDC2011T07

category	publisher	doc(%)	category	publisher	doc(%)
extreme-right	thegatewaypundit.com	94.44	left	salon.com	100.00
extreme-right	dcclothesline.com	85.71	left	gq.com	60.00
extreme-left	trueactivist.com	62.50	left	rawstory.com	40.00
right	pjmedia.com	100.00	left	opednews.com	100.00
right	express.co.uk	36.84	left	people.com	20.00
right	opslens.com	100.00	right-center	realclearpolitics.com	92.86
right	insider.foxnews.com	27.27	right-center	circa.com	12.50
right	foxnews.com	50.00	left-center	cbsnews.com	11.11
right	washingtonexaminer.com	57.14	left-center	heavy.com	7.69
right	bizpacreview.com	40.00	left-center	nytimes.com	30.00
right	nypost.com	66.67	center	snopes.com	8.33
right	bearingarms.com	66.67	center	nfl.com	0.00

Table 1: Selected publishers with their bias categories and percentage of biased articles in the dataset.

Features	Classifier	
	LR	SVM
BoW (without (c))	58.83	59.72
BoW	60.67	**60.93**
BoW-binary	60.61	60.68
BoW-TFIDF (without (c))	60.15	59.61
BoW-TFIDF	60.86	60.90
N-grams	60.73	59.13
N-grams-binary	60.18	59.74
N-grams-TFIDF (without (c))	59.65	59.72
N-grams-TFIDF	60.51	60.61

Table 2: Validation accuracy after fine-tuning. Without (c) means that the training set is not cleaned with the pre-processing step (c). Cleaning helps improve accuracy.

Features	Dim.	LR	SVM
GloVe	100	72.45	78.33 (rbf)
GloVe	200	72.76	76.78 (rbf)
GloVe	300	72.45	**79.57** (rbf)

Table 3: Accuracy of different GloVe vector dimensions.

Features	Dim.	LR	SVM
TFIDF	50K	77.09	77.71 (linear)
GloVe	300	72.45	**79.57** (rbf)
GloVe + Feat	327	75.85	78.33 (rbf)
Doc2Vec	400	71.83	78.95 (rbf)
Doc2Vec + Feat	427	77.71	75.85 (rbf)

Table 4: Accuracy of our model that is trained using training-2. The majority baseline is 63% accuracy.

task) that is labeled by crowd workers. Since the additional feature set does not improve the performance, it is not further evaluated. In Table 5, the accuracy, precision, recall, and F1-score on the held-out test set are shown. Our classifiers tend to have a higher false negative rate. This can be due to the imbalance in the training data. Further experiments would be required to see whether re-sampling to have a balanced training set can improve that.

Features	Acc.	Precision	Recall	F1
TFIDF	74.36	80.00	64.97	71.70
GloVe	**80.57**	85.82	73.25	79.04
Doc2Vec	73.89	82.61	60.51	69.85

Table 5: Submission results on the held-out test set, with metrics including accuracy, precision, recall, and F1-score.

5 Conclusion and Future Work

In this paper, we present the system we use to compete in the SemEval-2019 hyperpartisan news detection task. The final model we use is a kernel SVM trained with pre-trained GloVe vectors. It turns out that a simple method which requires the least training time performs the best in this case.

Both system 1 and system 2 have interesting future work to be done. For system 1, it is interesting to correct the labels or filter the articles in order to obtain a cleaner data to learn from. For system 2, we plan to use contextual embeddings (Peters et al., 2018) or pre-trained language models (Radford, 2018; Devlin et al., 2018) to extract representations that are then fed into downstream classifiers. The high performances of the models made them interesting to compare with.

References

Mark Daniel Bernhardt, Stefan Krasa, and Mattias K Polborn. 2008. Political polarization and the electoral effects of media bias. *Journal of Public Economics*, 92(5-6):1092–1104.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 855–864, New York, NY, USA. ACM.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Vivek Kulkarni, Junting Ye, Steve Skiena, and William Yang Wang. 2018. Multi-view models for political ideology detection of news articles. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3518–3527. Association for Computational Linguistics.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1188–II–1196. JMLR.org.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers.

Subhabrata Mukherjee and Gerhard Weikum. 2015. Leveraging joint interactions for credibility analysis in news communities. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 353–362, New York, NY, USA. ACM.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 231–240. Association for Computational Linguistics.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1650–1659. Association for Computational Linguistics.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

Sida Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 90–94, Stroudsburg, PA, USA. Association for Computational Linguistics.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 347–354, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489. ACL.

UBC-NLP at SemEval-2019 Task 4: Hyperpartisan News Detection With Attention-Based Bi-LSTMs

Chiyu Zhang **Arun Rajendran** **Muhammad Abdul-Mageed**
Natural Language Processing Lab
The University of British Columbia
chiyu94@alumni.ubc.ca, arun95@math.ubc.ca, muhammad.mageeed@ubc.ca

Abstract

We present our deep learning models submitted to the SemEval-2019 Task 4 competition focused at Hyperpartisan News Detection. We acquire best results with a Bi-LSTM network equipped with a self-attention mechanism. Among 33 participating teams, our submitted system ranks top 7 (65.3% accuracy) on the *labels-by-publisher* sub-task and top 24 out of 44 teams (68.3% accuracy) on the *labels-by-article* sub-task (65.3% accuracy). We also report a model that scores higher than the 8th ranking system (78.5% accuracy) on the *labels-by-article* sub-task.

1 Introduction

Spread of *fake news* (e.g., Allcott and Gentzkow (2017); Horne and Adali (2017)) (or 'low-quality' information (Qiu et al., 2017), among other terms) can have destructive economic impacts (Sandoval, 2008), result in dangerous real world consequences (Akpan, 2016), or possibly undermine the very democratic bases of modern societies (Qiu et al., 2017; Allcott and Gentzkow, 2017). Several approaches have been employed for detecting fake stories online, including detecting the sources that are highly polarized (or *hyperpartisan*)] (Potthast et al., 2017). Detecting whether a source is extremely biased for or against a given party can be an effective step toward identifying fake news.

Most research on news orientation prediction employed machine learning methods based on feature engineering. For example, Pla and Hurtado (2014) use features such as text n-grams, part-of-speech tags, hashtags, etc. with an SVM classifier to tackle political tendency identification in twitter. Potthast et al. (2017) investigate the writing style of hyperpartisan and mainstream news using a random forest classifier (Koppel et al., 2007). Further, Preoţiuc-Pietro et al. (2017) use a linear regression algorithm to categorize Twitter users into a fine-grained political group. The authors were able to show a relationship between language use and political orientation.

Nevertheless, previous works have not considered the utility of deep learning methods for hyperpartisanship detection. Our goal is to bridge this gap by investigating the extent to which deep learning can fare on the task. More precisely, we employ several neural network architectures for hyperpartisans news detection, including long short-term memory networks (LSTM), convolutional neural networks (CNN), bi-directional long short term memory networks (Bi-LSTM), convolutional LSTM (CLSTM), recurrent convolutional neural network (RCNN), and attention-based LSTMs and Bi-LSTMs.

We make the following contributions: (1) we investigate the utility of several deep learning models for classifying hyperpartisan news, (2) we test model performance under a range of training set conditions to identify the impact of training data size on the task, and (3) we probe our models with an attention mechanism coupled with a simple visualization method to discover meaningful contributions of various lexical features to the learning task. The rest of the paper is organized as follows: data are described in Section 2, Section 3 describes our methods, followed by experiments in Section 4. Next, we explain the results in detail and our submission to SemEval-2019 Task4 in Section 4. We present attention-based visualizations in Section 5, and conclude in Section 6.

2 Data

Hyperpartisan news detection is the SemEval-2019 task 4 (Kiesel et al., 2019). The task is set up as binary classification where data released by organizers are labeled with the tagset

	Labels-by-Publisher				Labels-by-Article		
	Train	**Dev**	**Test**	**Total**	**Train**	**Test**	**Total**
Hyperpartisan	383,151	66,849	50,000	500,000	214	24	238
Non- Hyperpartisan	416,849	33,151	50,000	500,000	366	41	407
Total	800,000	100,000	100,000	1,000,000	580	65	645

Table 1: Distribution of labels over our data splits.

{*hyperpartisan, not-hyperpartisan*}. The dataset has two parts, pertaining how labeling is performed. For **Part 1: labels-by-publisher**, labels are propagated from the publisher level to the article level. Part 1 was released by organizers twice. First 1M articles (less clean) were released, but then 750K (cleaner, de-duplicated) articles were released. We use all the 750K articles but we also add 250K from the first release, ensuring there are no duplicates in the articles and we also perform some cleaning of these additional 250K articles (e.g., removing error symbols). We ensure we have the balanced classes {hyperpartisan, not-hyperpartisan}, with 500K articles per class. For experiments, we split Part 1 into 80% train, 10% development (dev), and 10% test.

The labeling method for Part 1 assumes all articles by the same publisher will reflect the publisher's same polarized category. This assumption is not always applicable, since some articles may not be opinion-based. For this reason, organizers also released another dataset, **Part 2: labels-by-article**, where each individual article is assigned a label by a human. Part 2 is smaller, with only 645 articles (238 hyperpartisan and 407 non-hyperpartisan). Since Part 2 is smaller, we split it into 90% train and 10% test. Since we do not have a dev set for Part 2, we perform all our Hyperparameter tuning on the Part 1 dev set exclusively. Table 1 shows the statistics of our data.

3 Methods

3.1 Pre-processing

We lowercase all the 1M articles, tokenize them into word sequences, and remove stop words using *NLTK* [1]. For determining parameters like maximum sequence length and vocabulary size, we analyze the 1M articles, and find the number of total tokens to be 313,257,392 and the average length of an article to be 392 tokens (with a standard deviation of 436 tokens), and the number of types (i.e., unique tokens) to be 773,543. We thus set the maximal length of sequence in our models to be 392, and choose an arbitrary (yet reasonable) vocabulary size of 40,000 words.

[1]https://www.nltk.org/

3.2 Architectures

Deep learning has boosted performance on several NLP tasks. For this work, we experiment with a number of methods that have successfully been applied to text classification. Primarily, we employ a range of variations and combinations of recurrent neural networks (RNN) and convolutional neural networks (CNN). RNNs are good summarizers of sequential information such as language, yet suffer from gradient issues when sequences are very long. Long-Short Term Memory networks (LSTM) (Hochreiter and Schmidhuber, 1997) have been proposed to solve this issue, and so we employ them. Bidirectional LSTM (Bi-LSTM) where information is summarized from both left to right and vice versa and combined to form a single representation has also worked well on many tasks such as named entity recognition (Limsopatham and Collier, 2016), but also text classification (Abdul-Mageed and Ungar, 2017; Elaraby and Abdul-Mageed, 2018). As such, we also investigate Bi-LSTMs on the task. Attention mechanism has also been proposed to improve machine translation (Bahdanau et al., 2014), but was also applied successfully to various other tasks such as speech recognition, image captioning generation, and text classification (Xu et al., 2015; Chorowski et al., 2015; Baziotis et al., 2018; Rajendran et al., 2019). We employ a simple attention mechanism (Zhou et al., 2016b) to the output vector of the (Bi-)LSTM layer. Although CNNs have initially been proposed for image tasks, they have also been shown to work well for texts (e.g., (Kim, 2014)) and so we employ a CNN. In addition, neural network architectures that combine different neural network architectures have shown their advantage in text classification (e.g., sentiment analysis). For example, improvements on text classification accuracy were observed applying a model built on a combination of Bi-LSTM and two-dimensional CNN (2DCNN) compared to separate RNN and CNN models (Zhou et al., 2016a). Moreover, a combination of CNN and LSTM (CLSTM) outperform both CNN and LSTM on sentiment classification and question classification tasks (Zhou et al., 2015). The experiments of Lai et al. (2015) demonstrate that recurrent convolutional neural networks (RCNNs) outperforms CNN and RNN on text classification. For these reasons, we also experiment with RCNN and CLSM.

3.3 Hyper-Parameter Optimization

For all our models, we use the top 40K words from Part 1 training set (*labels-by-publisher*) as our vocabulary. We initialize the embedding layers with Google News Word2Vec model. [2] For all networks, we use a single hidden layer. We use dropout (Srivastava et al., 2014) for regularization.

Models	**Hidden No.**	**Drop out**	**Kernel size**	**Kernel No**
LSTM	300	0.1	N/A	N/A
Bi-LSTM	200	0.0	N/A	N/A
LSTM+Attn	500	0.0	N/A	N/A
Bi-LSTM+Attn	500	0.0	N/A	N/A
CNN	N/A	0.1	[4,5,6]	200
RCNN	200	0.3	N/A	N/A
CLSTM	200	0.3	[2,3,4]	70

Table 2: Our best Hyper-parameters.

For the best Hyper-parameters for each network, we use the Part 1 dev set to identify the number of units (between 100 and 600) in each network's hidden layer and the dropout rate (choosing values between 0 and 1, with 0.1 increments). For the CNNs (and their variations), we use 3 kernels with different sizes (with groups like 2,3,4) and identify the best number of kernel filters (between 30 to 300). All Hyper-parameters are identified using the Part 1 dev set. Table 2 presents the detailed optimal Hyper-parameters for all our models. [3]

[2]`https://github.com/mmihaltz/word2vec-GoogleNews-vectors`

[3]For all our networks, we identify our best learning rate as 0.001. For this reason, we do not provide learning rate in Table 2.

4 Experiments & Results

We run two main sets of experiments, which we will refer to as EXP-A and EXP-B. For EXP-A, we train on the labels-by-publisher (Part 1) train set, tune on dev, and test on test. All related results are reported in Table 3. As Table 3 shows, our best macro F_1 as well as accuracy is acquired with Bi-LSTM with attention (Bi-LSTM+ATTN). For EXP-B, we use Part 1 and Part 2 datasets in tandem, where we train on each train set independently and (1) test on its test data, but also (2) test on the other set's test data. We also (3) fine-tune the models pre-trained on the bigger dataset (Part 1) on the smaller dataset (Part 2), to test the transferrability of knowledge from these bigger models. Related results (only in accuracy, for space) are in Table 4. Again, the best accuracy is obtained with Bi-LSTM with attention.

SemEval-2019 Task 4 Submissions: We submitted our Bi-LSMT+Attention model from EXP A to the *labels-by-publisher leaderboard* in TIRA (Potthast et al., 2019), and it ranked top 7 out of the 33 teams, scoring at *accuracy=0.6525* on the competition test set. [4] From EXP-B, we submitted our model based on Bi-LSMT+Attention that was trained on Part 2 train exclusively dataset (by-ATC in Table 4) to the *labels-by-article leaderboard*. It ranked top 24th out of 44 teams (*accuracy=0.6831*). Post-competition, we submitted our EXP-B model that is pre-trained on the by-publisher data and fine-tuned on the by-article data (by-PSH+by-ATC in Table 4) to the *labels-by-article leaderboard*. It ranked top 8th, with *78.50% accuracy*. This might be due to the ability of this specific model to transfer knowledge from the big (*by-publisher*) training set to the smaller (*by-article*) data (i.e., better generalization).

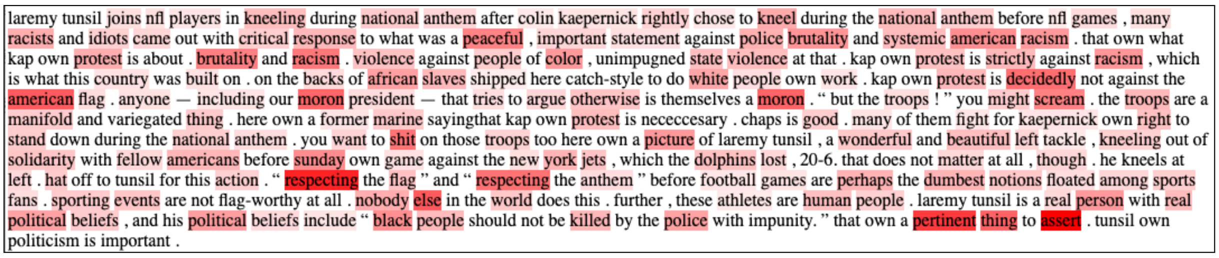
5 Attention Visualization

For better interpretation, we present a visualization of words of our best model from EXP-B (by-PSH+by-ATC in Table 4) attends to across the two classes, as shown in Figure 1. The color intensity in the Figure corresponds to the weight given to each word by the self-attention mechanism and signifies the importance of the word for final prediction. As shown in Figure 1 (a), some heavily polarized terms such as 'moron', 'racism', 'shit',

[4]The competition test set is different from our own test set, which we created by splitting the data we received.

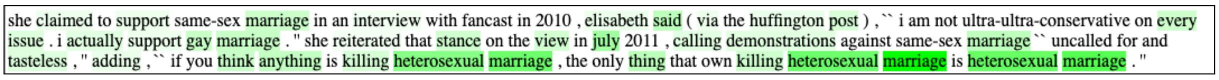
Models	Test Accuracy	Precision		Recall		F_1	
		Hyper	Non-hyper	Hyper	Non -hyper	Hyper	Non-hyper
LSTM	0.9174	0.8927	0.9422	**0.9392**	0.8977	0.9154	0.9203
CNN	0.9147	0.9179	0.9115	0.9121	0.9173	0.9150	0.9114
Bi-LSTM	0.9196	0.9097	0.9295	0.9281	0.9114	0.9188	0.9203
LSTM+ATTN	0.9071	0.8755	0.9388	0.9347	0.8829	0.9041	0.9100
Bi-LSTM+ATTN	**0.9368**	**0.9493**	0.9262	0.9347	**0.9480**	**0.9376**	**0.9360**
CLSTM	0.8977	0.9181	0.8773	0.9147	0.8821	0.8956	0.8998
RCNN	0.9161	0.9380	0.8946	0.8972	0.9364	0.9171	0.9150
Random Forest	0.7723	0.5312	**0.9456**	0.8824	0.7333	0.6628	0.8260

Table 3: Performance of Predicting Hyperpartisan News (EXP-A).



laremy tunsil joins nfl players in kneeling during national anthem after colin kaepernick rightly chose to kneel during the national anthem before nfl games , many racists and idiots came out with critical response to what was a peaceful , important statement against police brutality and systemic american racism . that own what kap own protest is about . brutality and racism . violence against people of color , unimpugned state violence at that . kap own protest is strictly against racism , which is what this country was built on . on the backs of african slaves shipped here catch-style to do white people own work . kap own protest is decidedly not against the american flag . anyone — including our moron president — that tries to argue otherwise is themselves a moron . “ but the troops ! ” you might scream . the troops are a manifold and variegated thing . here own a former marine sayingthat kap own protest is nececcesary . chaps is good . many of them fight for kaepernick own right to stand down during the national anthem . you want to shit on those troops too here own a picture of laremy tunsil , a wonderful and beautiful left tackle , kneeling out of solidarity with fellow americans before sunday own game against the new york jets , which the dolphins lost , 20-6. that does not matter at all , though . he kneels at left . hat off to tunsil for this action . “ respecting the flag ” and “ respecting the anthem ” before football games are perhaps the dumbest notions floated among sports fans . sporting events are not flag-worthy at all . nobody else in the world does this . further , these athletes are human people . laremy tunsil is a real person with real political beliefs , and his political beliefs include “ black people should not be killed by the police with impunity. ” that own a pertinent thing to assert . tunsil own politicism is important .

(a) Hyperpartisan.



she claimed to support same-sex marriage in an interview with fancast in 2010 , elisabeth said (via the huffington post) , `` i am not ultra-ultra-conservative on every issue . i actually support gay marriage . " she reiterated that stance on the view in july 2011 , calling demonstrations against same-sex marriage `` uncalled for and tasteless , " adding , `` if you think anything is killing heterosexual marriage , the only thing that own killing heterosexual marriage is heterosexual marriage . "

(b) Non-hyperpartisan.

Figure 1: Attention heat-map for article examples.

	Test on	Train on		
		by-PSH	by-ATC	by-PSH +by-ATC
LSTM	**by-PSH**	0.9174	0.5331	0.8369
	by-ATC	0.5917	0.7833	0.7667
BiLSTM	**by-PSH**	0.9196	0.5562	0.8089
	by-ATC	0.5783	0.6540	0.7833
LSTM+A	**by-PSH**	0.9071	0.7397	0.8509
	by-ATC	0.5783	0.8166	0.7833
BiLSTM+A	**by-PSH**	**0.9368**	0.5412	0.7908
	by-ATC	0.5504	**0.8615**	**0.8153**

Table 4: Results with Part 1 and Part 2 datasets (EXP-B). Last column "by-PSH +by-ATC" is the setting of our models pre-trained on Part 1 and fine-tuned on Part 2. +A= added attention.

'scream', and 'assert' are associated with the hyperpartisan class. It is clear from the content of the article from which the example is drawn that it is a highly opinionated article. In Figure 1 (b), items such as 'heterosexual marriage', 'gay', 'July', and 'said' carry more weight than other items. These items are not as much opinionated as those in 1 (a), and some of them (e.g., 'July' and 'said') are more of factual and reporting devices than mere carriers of ad hominem attacks. These features show that some of the model attentions are meaningful.

6 Conclusion

In this paper, we described our system of hyperpartisan news detection to the 4th SemEval-2019 shared task. Our best models are based on a Bi-LSTM with self-attention. To understand our models, we also visualize their attention weights and find meaningful patterns therein.

7 Acknowledgement

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), the Social Sciences Research Council of Canada (SSHRC), WestGrid (`www.westgrid.ca`), and Compute Canada (`www.computecanada.ca`).

References

Muhammad Abdul-Mageed and Lyle Ungar. 2017. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 718–728.

Nsikan Akpan. 2016. The very real consequences of fake news stories and why our brain cant ignore them. *PBS News Hour*.

Hunt Allcott and Matthew Gentzkow. 2017. Social media and fake news in the 2016 election. Technical report, National Bureau of Economic Research.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Christos Baziotis, Nikos Athanasiou, Alexandra Chronopoulou, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, Shrikanth Narayanan, and Alexandros Potamianos. 2018. Ntua-slp at semeval-2018 task 1: Predicting affective content in tweets with deep attentive rnns and transfer learning. *arXiv preprint arXiv:1804.06658*.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in neural information processing systems*, pages 577–585.

Mohamed Elaraby and Muhammad Abdul-Mageed. 2018. Deep models for arabic dialect identification on benchmarked data. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, pages 263–274.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Benjamin D Horne and Sibel Adali. 2017. This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. *arXiv preprint arXiv:1703.09398*.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Moshe Koppel, Jonathan Schler, and Elisheva Bonchek-Dokow. 2007. Measuring differentiability: Unmasking pseudonymous authors. *Journal of Machine Learning Research*, 8(Jun):1261–1276.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273.

Nut Limsopatham and Nigel Collier. 2016. Learning orthographic features in bi-directional lstm for biomedical named entity recognition. In *Proceedings of the Fifth Workshop on Building and Evaluating Resources for Biomedical Text Mining (BioTxtM2016)*, pages 10–19.

Ferran Pla and Lluís-F Hurtado. 2014. Political tendency identification in twitter using sentiment analysis techniques. In *Proceedings of COLING 2014, the 25th international conference on computational linguistics: Technical Papers*, pages 183–192.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. TIRA Integrated Research Architecture. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2017. A stylometric inquiry into hyperpartisan and fake news. *arXiv preprint arXiv:1702.05638*.

Daniel Preoţiuc-Pietro, Ye Liu, Daniel Hopkins, and Lyle Ungar. 2017. Beyond binary labels: political ideology prediction of twitter users. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 729–740.

Xiaoyan Qiu, Diego FM Oliveira, Alireza Sahami Shirazi, Alessandro Flammini, and Filippo Menczer. 2017. Limited individual attention and online virality of low-quality information. *Nature Human Behavior*, 1:0132.

Arun Rajendran, Chiyu Zhang, and Muhammad Abdul-Mageed. 2019. Happy together: Learning and understanding appraisal from natural language. In *Proceedings of the AAAI2019 Second Affective Content Workshop (AffCon 2019)*, pages 00–00.

Greg Sandoval. 2008. Whos to blame for spreading phony jobs story? *CNet News*, pages 4–46.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1785–1794.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016a. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639*.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016b. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212.

Vernon-fenwick at SemEval-2019 Task 4: Hyperpartisan News Detection using Lexical and Semantic Features

Vertika Srivastava **Ankita Gupta** **Divya Prakash**
Sudeep Kumar Sahoo **Rohit R.R** **Yeon Hyang Kim**

Samsung R&D Institute India, Bangalore
{v.srivastava, gupta.ankita, p.divya, sudeep.sahoo, rohit.r.r, purine.kim}@samsung.com

Abstract

In this paper, we present our submission for SemEval-2019 Task 4: Hyperpartisan News Detection. Hyperpartisan news articles are sharply polarized and extremely biased (one-sided). It shows blind beliefs, opinions and unreasonable adherence to a party, idea, faction or a person. Through this task, we aim to develop an automated system that can be used to detect hyperpartisan news and serve as a prescreening technique for fake news detection. The proposed system jointly uses a rich set of handcrafted textual and semantic features. Our system achieved 2nd rank on the primary metric (82.0% accuracy) and 1st rank on the secondary metric (82.1% F1-score), among all participating teams. Comparison with the best performing system on the leaderboard[1] shows that our system is behind by only 0.2% absolute difference in accuracy.

1 Introduction

Today in the age of digitization, a smartphone has become an indispensable tool for information sharing and consumption. It is much more convenient for users to read news through online articles and social media platforms. These platforms provide them quick and easy access to information almost everywhere. However, it is highly possible that the shared information is unverified and may bias the reader's opinions. This issue is exacerbated by the fact that most of the people find it difficult to distinguish between what's real and objective, what's fake and what's partisan.

Although these online news articles are expected to be written well-balanced without any prejudices, the authors/media houses may at times spill their standpoints and beliefs. Instead of providing a holistic view to the readers, the author tries to convey a picture with which he agrees and thus making the audience biased towards a party or a faction. When these news articles are extremely polarized towards one side of the argument, they are referred to as "Hyperpartisan News Articles".

This extreme polarization can leave users vulnerable to detrimental arguments and cloud their judgment to make objective decisions. These hyperpartisan articles may also carry some common elements of fake news. They are typically used to spread propaganda and manipulate readers. It targets human psychology by creating confirmation bias and echo chambers and therefore impairing their ability to dispel the hyperpartisan articles in favor of neutral articles.

SemEval-2019 Task 4 aims to solve this issue of hyperpartisanship. The objective of this task is to detect if a given news article has hyperpartisan arguments (extremely one-sided). In this paper, we have described our system that automates the process of identifying and annotating news articles as hyperpartisan or not.

2 Related Work

The problem of fake news and hyperpartisan has been discussed earlier by Potthast et al. (2017). They followed a style based approach to tackle the problem and have also suggested that writing style of left-wing and right-wing news are quite similar. Apart from this, there hasn't been much work in hyperpartisan news detection. Our approach is inspired by some recent work in the domain of sentiment analysis (Pontiki et al., 2016) and bias detection (Patankar et al., 2018; Recasens et al., 2013; Patankar and Bose, 2017; Baly et al., 2018). Jian and Wilson (2018) have explored linguistic signals embedded in news articles. They have used these linguistic clues to detect the spread of misinformation via social media. Iyyer et al. (2014) have

[1] https://pan.webis.de/semeval19/semeval19-web/leaderboard.html

studied the impact of words in identifying people's ideology and have proposed RNN to capture semantic features of a sentence.

3 System Description

Our hyperpartisan news detection system consists of three phases: 1) preprocessing, 2) generating article representation, and 3) training a classifier. An overview of our system is shown in Figure 1.

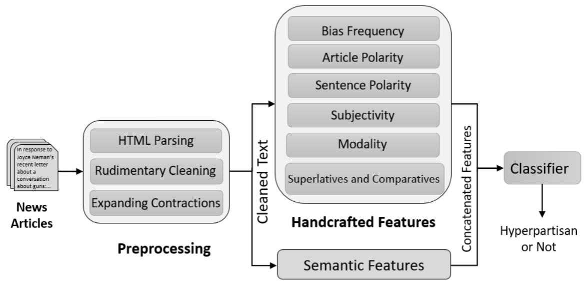


Figure 1: System Pipeline.

3.1 Preprocessing

The original dataset consisted of news articles along with HTML tags. Hyperpartisan news detection cleaner[2] was used to convert original text to plain text by removing tags. In the next step, further rudimentary cleaning of articles was done. We have also expanded contractions in the dataset like 'shan't' was converted to "shall not", 'don't' to "do not" etc.

3.2 Article Representation

Hyperpartisan news articles are extremely one sided and exhibits blind beliefs as compared to a neutral news articles. We have observed that writers of such articles generally, manifest usage of harsh tone and inflammatory language, they even exaggerate and convey opinions to stress their ideology. Hyperpartisan news articles tend to use superlatives and comparatives frequently to dramatize or exaggerate situations.

Polarity at an article level can capture emotions and sentiments of the article, it may also capture contextual polarity. Polarity at sentence level helps to identify bias localized to a sentence which might not be perceptible at an article level. It helps to shift the focus on bias-heavy sentences. By combining polarity at both levels, we can capture the tone, overpraise and sentiment in an article. Subjectivity, modality and bias lexicons help to discover the attitude, prejudice and beliefs expressed by the author. Building from these insights, we created a set of handcrafted textual features (HF), which are based on writing style, linguistics, and lexicons.

The problem with handcrafted features is that they don't capture the semantic relationship among the sentences. To solve this problem, we incorporated semantic features (SF) that can capture long-range dependencies of sentences and bring out the semantics of the article. The drawback of semantic features obtained via word based embeddings like Glove, is that it ignores word sequencing. In our approach, we have also explored features generated via distributed document representation (Universal Sentence Encoder or Doc2Vec) that are agnostic to word orderings and captures the semantics of an article.

Consider a set of N news articles $A = \{a_1...a_N\}$. Each article a_i has a set of $S = \{s_1...s_m\}$ sentences and a set of $W = \{w_1...w_l\}$ words, where l is the length of the article. We jointly used HF and SF to obtain article representation ($ArtRep$), where $\oplus$ is concatenation operator:

$$ArtRep = HF \oplus SF \tag{1}$$

Handcrafted features used in our system is described in Section 4 and semantic features are discussed in Section 5.

4 Handcrafted Features

Bias Score: For identifying bias words in an article, bias lexicon built from NPOV corpus of Wikipedia articles (Recasens et al., 2013) was used. Wikipedia advocates Neutral Point of View policy (NPOV), articles falling under NPOV dispute category were used to build this corpus. Bias score is the frequency of article words that occur in bias lexicon.

Article Level Polarity: Polarity of an article ($APol$) was extracted using MPQA Subjectivity lexicon (Wilson et al., 2005) ($SLex$), which lists around 8000 words with their prior polarity and their subjectivity type. Let a set of prior positive polarity words in an article be $PLex_i$ and negative polarity words be $NLex_i$. We computed positive ($APol_i^+$) and negative polarity score ($APol_i^-$) of

[2]https://github.com/webis-de/semeval19-hyperpartisan-news-detection-article-cleaner

an article a_i, where $\mathbb{1}$ is an indicator function:

$$APol_i^+ = \frac{1}{l}\sum_{j=1}^{l} \mathbb{1}(w_j \in (PLex_i \cap SLex)) \quad (2)$$

$$APol_i^- = \frac{1}{l}\sum_{j=1}^{l} \mathbb{1}(w_j \in (NLex_i \cap SLex)) \quad (3)$$

Sentence Level Polarity: Polarity was further fine-grained to generate features for sentences of a news article using Pattern toolkit for English [3]. A sentence s_j was given as an input to the toolkit and a polarity score α_j in the range of [-1.0, 1.0] was obtained. Positive ($PolScore_i^+$), negative ($PolScore_i^-$) and neutral polarity score ($PolScore_i^{Neu}$) of an article a_i was computed with $|0.1|$ as a threshold as it gave the best results for our system:

$$PolScore_i^+ = \sum_{j=1}^{m} \mathbb{1}(\alpha_j > 0.1) \quad (4)$$

$$PolScore_i^- = \sum_{j=1}^{m} \mathbb{1}(\alpha_j < -0.1) \quad (5)$$

$$PolScore_i^{Neu} = \sum_{j=1}^{m} \mathbb{1}(-0.1 \leq \alpha_j \leq 0.1) \quad (6)$$

Subjectivity and Modality: Subjectivity score is computed using Sentiment module of Pattern toolkit for English[3]. Toolkit gives a score based on adjectives and their context in the range of [0.0, 1.0]. Modality is a measure of the degree of certainty. It was computed using Modality module of Pattern toolkit for English[3].

Superlatives and Comparatives : Intensifying lexicons like adjectives and adverbs in superlative and comparative degree were used. We ran POS tagger from NLTK (Bird and Loper, 2004) on the article text to identify Subjective and Comparative adjectives and adverbs and their corresponding frequencies in the text were used as a feature.

5 Semantic Features

5.1 Glove

Glove (Pennington et al., 2014) provides distributional vector representations of words in the semantic space. We have used 300-dimensional Glove embeddings trained on Common Crawl data of 2.2 million words and 840 billion tokens. An article was tokenized into sentences and further into words to obtain it's article representation. Each of these words was vectorized using Glove pre-trained embeddings. Article representation was generated by averaging (Wieting et al., 2015) these 300-dimensional word embeddings.

[3]https://www.clips.uantwerpen.be/pages/pattern-en

Dataset	Articles
Hyperpartisan	238
Non-Hyperpartisan	407

Table 1: ByArticle Dataset statistics.

5.2 Doc2Vec

Doc2Vec (D2V) (Le and Mikolov, 2014) is an unsupervised algorithm to learn distributed representation of multi-word sequences in semantic space. We have used Python implementation of Doc2Vec provided by gensim to learn embeddings for the news articles. For our experiments, we have used 512-dimensional embeddings generated from D2V on article text.

5.3 Universal Sentence Encoder

Universal Sentence Encoder (USE) (Cer et al., 2018) is a pretrained model to generate embeddings for sentences, phrases and much larger multi-word sequences. It has shown good performance on diverse NLP tasks for e.g., phrase level opinion extraction and sentiment classification. USE takes English text as an input and generates 512-dimensional embedding. In our best system, we have used these 512-dimensional article embeddings, generated on feeding article text to USE.

6 Experiments

6.1 Dataset

We have used *ByArticle* dataset provided in the task, which is labeled through crowdsourcing. Dataset consists of 645 news articles and a label to denote if it is hyperpartisan or not, details of the dataset are provided briefly in Table 1. More information on the dataset can be found in (Kiesel et al., 2019).

Model	Acc.	Prec.	Recall	F1
Baseline	46.18	46.03	44.27	45.13
1st ranked system	**82.17**	**87.13**	75.48	80.90
SM	58.76	58.51	60.19	59.34
SC	65.76	70.37	54.46	61.40
B	68.63	73.31	58.60	65.13
SP	68.63	72.24	60.51	65.86
AP	65.13	64.90	65.92	65.40
HF	70.22	72.76	64.65	68.47
D2V+HF	73.41	69.60	83.12	75.76
Glove+HF	78.34	82.01	72.61	77.03
USE+HF	82.01	81.50	**82.80**	**82.15**

Table 2: SemEval-2019 Task 4 Performance comparison on hidden test data (SM: Subjectivity and Modality, SC: Superlatives and Comparatives, B: Bias Score, AP: Article Level Polarity, SP: Sentence Level Polarity, HF: set of all Handcrafted Features). All results reported in the table are in percentage and rounded to 2 decimal places.

6.2 Results and Analysis

Handcrafted features were concatenated with semantic features, to generate a rich article representation which was fed to the classifier as an input. A L2-regularized logistic regression (Pedregosa et al., 2011) classifier was trained with 10-fold cross-validation. Since the training dataset is unbalanced, we have used class weighted logistic regression by weighing classes inversely proportional to their frequency.

For evaluation, a balanced hidden test data (*ByArticle*) was provided by the organizers through TIRA (Potthast et al., 2019). We have used accuracy for performance evaluation, which was a primary performance measure in the task. Apart from this, we have also reported precision, recall, and F1-score on the dataset.

Table 2 shows the performance of our various approaches against the baseline results (Task 4 semeval-pan-2019-baseline on TIRA[4]). From the results, we can observe that our best performing system has outperformed the baseline by a huge margin, recording an absolute jump of 35.83% accuracy. We can also see that our system is as good as the best system (bertha-von-suttner[4]) submitted for the task, which has 82.17% accuracy, showing that we were behind by only 0.16%.

In order to assess the importance of each handcrafted feature, we performed experiments by using individual features and their combination (HF). In fact, Table 2 highlights that all the features jointly perform well with 70.22% accuracy. We found that bias lexicon and polarity based features are the most informative ones.

In an attempt to further improve our model's performance, we experimented by combining semantic features with handcrafted features as described in Section 3.2. Our results have also shown that handcrafted features alone aren't sufficient and better performance can be achieved by combining them with semantic features. USE+HF has beaten all our other models and showed an absolute improvement of 11.79% accuracy over the HF model. It also attained 1st rank on the leaderboard (based on F1-score) and 2nd overall rank (based on Accuracy).

The poor performance of D2V+HF can be attributed to the training of D2V as typically D2V requires large data for training. Results have also shown that USE+HF performed better than Glove+HF, and thus validating our earlier claim of limitations of word-based embeddings.

7 Conclusion and Future Work

In this paper, we proposed a novel approach to detect hyperpartisan arguments in a news article. Our system ranked 2nd in SemEval-2019 Task 4. Our approach leverages rich semantic and handcrafted textual features. In the paper, we have also studied the importance of capturing semantic relationship among sentences of an article. Our system employed linguistic and lexical features to detect polarity, sentiments and blind beliefs exhibited in an article. Experiments with various model configurations demonstrated the effectiveness of our approach.

Detecting hyperpartisan in news articles should also, involve incorporation of world knowledge, as statements as an individual may not be extremely biased but when seen from a global perspective, they turn out to be hyperpartisan. As a future work, we would like to exploit the use of external knowledge. We would also like to investigate the role of credibility of news sources (Baly et al., 2018; Popat et al., 2018) in detecting hyperpartisan news articles.

[4]https://www.tira.io/task/hyperpartisan-news-detection/dataset/pan19-hyperpartisan-news-detection-by-article-test-dataset-2018-12-07/

References

Ramy Baly, Georgi Karadzhov, Dimitar Alexandrov, James Glass, and Preslav Nakov. 2018. Predicting factuality of reporting and bias of news media sources. *arXiv preprint arXiv:1810.01765*.

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1113–1122.

Shan Jiang and Christo Wilson. 2018. Linguistic signals under misinformation and fact-checking: Evidence from user comments on social media. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):82.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Anish Anil Patankar and Joy Bose. 2017. Bias discovery in news articles using word vectors. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 785–788. IEEE.

Anish Anil Patankar, Joy Bose, and Harshit Khanna. 2018. A bias aware news recommendation system. *arXiv preprint arXiv:1803.03428*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, AL-Smadi Mohammad, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 19–30.

Kashyap Popat, Subhabrata Mukherjee, Andrew Yates, and Gerhard Weikum. 2018. Declare: Debunking fake news and false claims using evidence-aware deep learning. *arXiv preprint arXiv:1809.06416*.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. TIRA Integrated Research Architecture. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2017. A stylometric inquiry into hyperpartisan and fake news. *arXiv preprint arXiv:1702.05638*.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1650–1659.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics.

AndrejJan at SemEval-2019 Task 7: A Fusion Approach for Exploring the Key Factors pertaining to Rumour Analysis

Andrej Janchevski, Sonja Gievska
Faculty of Computer Science and Engineering
Ss. Cyril and Methodius University
Rugjer Boshkovikj 16, Skopje, Republic of North Macedonia
andrej.jancevski@students.finki.ukim.mk
sonja.gievska@finki.ukim.mk

Abstract

The viral spread of false, unverified and misleading information on the Internet has attracted a heightened attention of an interdisciplinary research community on the phenomenon. This paper contributes to the research efforts of automatically determining the veracity of rumourous tweets and classifying their replies according to stance. Our research objective was to investigate the interplay between a number of phenomenological and contextual features of rumours, in particular, we explore the extent to which network structural characteristics, metadata and user profiles could complement the linguistic analysis of the written content for the task at hand. The current findings strongly demonstrate that supplementary sources of information play significant role in classifying the veracity and the stance of Twitter interactions deemed to be rumourous.

1 Introduction

Social networks continue to contribute to the way people connect, stay informed and contribute above and beyond what concerns their lives. However, these platforms currently play a crucial role in viral spreading of false information, and no doubt will become even more instrumental in the future. Consequently, it will be increasingly important to devise systems and establish practices of automatically identifying, filtering and labeling false information in order to help people make sense of information dispersed through social channels.

The damaging consequences of malicious intents to misinform, confuse and provoke through social media platforms remain persistently high in the discussions of researchers and practitioners. Perhaps even more so than the benefits of information dissemination. While the veracity of some false information can be determined unambiguously from external sources, it is a major challenge to verify the truthfulness of rumourous postings. Indeed, predictive analysis remains the primary manner in which social platforms could face the challenge of identifying rumours and taking appropriate actions. Over the past several years, research has emerged and at the same time, many challenges remain.

Decoding elusive social phenomenon such as spreading rumours is challenging not only due to their complexity, but also due to the diversity of the underlying causes and heterogeneity in their manifestations. Rumours represent intertwined threads of sensemaking Bordia and Difonzo (2004) that are initiated and spread by people trying to explain, solve and remove uncertainty relating to events and persons that attract public interest Peterson and Gist (1951). We argue that solutions that address these multi-dimensional issues cannot solely rely on natural language processing (NLP). Combining natural language processing with social analytics extends beyond the traditional realms of either technology to a variety of emerging applications, including rumour analysis.

The analysis of rumours can take on different meanings depending on the application domain - this research focuses on identifying the veracity of a rumourous tweet (Task B) and the stance of its replies i.e., classifying responses to a rumourous post as supporting, denying, querying or commenting (SDQC) the claim (Task A), both part of the RumourEval 2019 Gorrell et al. (2019). While this work was developed on the foundations laid out by previous research in the field Derczynski et al. (2017), it is among the rare solutions that explore complementary types of information that could augment the linguistic analysis when analyzing rumourous tweets. After a brief discussion of relevant research that is closely related to our objective, we highlight the primary findings of our research.

2 Related work

Our research follows the line of work of the research groups that have contributed to the discourse on the rumour analysis with several experimental studies. A review of past SemEval related tasks points out that much of the research of the problem of examining the veracity and support for rumours focused more on language analysis and less on utilizing the external information made available by the Twitter dataset. The findings that extend across several studies is how rather simplistic NLP techniques and analysis are capable of obtaining satisfactory results when classifying the support of rumourous posting. The similarities and differences between the work presented in this paper and previous research is discussed.

A model for automatic identification of rumourous tweets and classification of their responses into two denying and supporting classes, presented in Qazvinian et al. (2011), is based on linguistic features, such as, unigrams, bigrams, POS tags, URLs, and hashtags. While, we built upon the experiences of this research, especially in the approach of replacing a multi-class classification problem with a hierarchical pipeline of binary classifiers, our approach differs in a number of aspects. In particular, we further enhance the representation of rumourous posts with a number of features, from word embeddings, sentiment and stylistic features to structural properties of interaction threads. In addition, we argue that a suitable preprocessing of tweets is essential for successive NLP steps, which is in contrast to their decision not to perform any preprocessing of the text.

The research study by Lukasik et al. (2015) extends on the work of Qazvinian et al. (2011), especially by incorporating preprocessing steps, bag-of-words (BoW) and word clustering. The approach presented in this paper follows some of these ideas, although it differs in the scope and the way features are operationalized. For instance, TF-IDF n-gram counts and clusters of Word2Vec embeddings were used in our model.

Based on the premise that psychological and sociological information could play a key role in determining the truthfulness of rumours and inspired by the research of Mihaylova et al. (2018) on fact checking of questions posted on online forums, we have further explored the effect of various metadata and information from users profiles on the performance of the task.

3 Methods

3.1 Dataset

The dataset used for evaluating the model proposed in this research was made available by Zubiaga et al. (2016b). The rumourous tweets and their corresponding interactions threads were harnessed in 2016, verified and labeled by journalists and sociologists using crowdsourcing platforms Zubiaga et al. (2016a). The dataset contains rumourous tweets written in English, associated with nine events, five being news stories and four concerning specific events. Out of a total of 4560 tweets, 297 represent rumourous posts annotated for veracity, while the rest are responses to the original rumour, annotated according to their stance i.e., supporting, denying, querying or commenting the initial claim.

3.2 Preprocessing

Tokenization, part-of-speech tagging, lemmatization, and substitutions were used as preprocessing techniques; the set and order of preprocessing steps varies between features. In addition, a number of context-appropriate corrections of language variations of English e-dialect were performed:

- Characters are converted to lower-case letters; for sentiment feature extraction the original letter case was preserved;
- URLs, numerical sequences, email addresses were replaced with special tokens (e.g., *URL*, *NUM*, *EMAIL*). Special tokens, *QUOTE* and *USER*, were used whenever the original rumour tweet was quoted or a user is mentioned in a response tweet;
- For each hashtag, the # sign is replaced with a special token *HASHTAG*, while the text of the hashtag was kept for further analysis;
- Emojis are identified and each was represented as a different token;
- Consecutive repetitions of a character in a word were contracted to 2 instances of the character;

- All special characters were removed, with the exception of `\s\n\r.,?!:-+` that were treated as separate tokens

3.3 Feature Extraction

Automatic extraction of information related to language, discourse and context is a difficult task. Among different combinations of features, the model that yielded the best result is described. Linguistic analysis for extraction of seven types of features was performed on the preprocessed text of each tweet. In addition, we have analyzed the content of user profiles, including profiles of the initiators of rumours and those replying to initial posts. Our assumption was that while tweet analysis captures indicators pertaining to a particular rumour thread, the language analysis of a user profile could provide insights into personality, attitudes and online behavior of a user.

Language style features - In examining previous research on rumour analysis, we found that stylistic features are frequently presented as simple statistical features which had an effect on the performance of the task. We have considered the following: number of words and sentences, average number of words per sentence, ratio of word vs. non-word tokens, percentage of present dictionary words, mean and variance of word length, and percentage of unique words. The same set of stylistic features were calculated for the content of each tweet as well the text content found in the profile of the user who has posted the tweet.

Language model n-grams - Unigrams, bigrams and trigrams were extracted from the tweets and users profiles, keeping only 1/8, 1/16 and 1/20 of the most frequent unique unigrams, bigrams and trigrams respectively. Six vocabularies were created; three n-gram vocabularies for the tweets and three vocabularies of n-grams found in user profiles. Term frequency-inverse document frequency (TF-IDF) values were calculated and utilized as the final language model features.

POS tags - In accordance with the well-established practice to complement the language model n-grams with their part-of-speech tags, TF-IDF values were calculated for two POS tag vocabularies, one relating to tweets, the other to self-descriptions left by users in their profiles.

Word2Vec embeddings - Stop-words were removed and lemmatization was performed on the tweets as well as user profile text. Two Word2Vec models were trained on the SemEval 2019 dataset, one was trained on the sentences of the tweets, the other on the content of user profiles. The dimensionality of the vectors was set to 500. A context window of size 5 was used, while words with frequencies above 0.001 were subsampled. The two vocabularies of embedding vectors were clustered using the K-Means algorithm; the parameter K was set automatically to ensure that each word cluster will contain an average of 10 items. At last, for each token list, a Bag-of-Centroids feature vector was calculated by counting the word clusters the tokens belong to.

Sentiment features - Consistent with related studies, which suggest the predictive power of affective words on the task relating to detecting deception in online text, we perform a polarity sentiment analysis on the tweets, calculating three polarity scores, positive, negative and neutral for each sentence. The NLTK, Vader Sentiment Intensity Analyzer, was used because of its reported robustness to the style of online e-language (e.g., capitalization, punctuation, slang) Hutto and Gilbert (2014). A sentiment feature vector for each tweet was generated from the mean and variance for each polarity score.

Network structure - Social network communication exchanges (e.g., tree-structured threads of tweets) are naturally represented as graphs (DAG) - nodes represent rumour posts and their responses, while directed edges of the graph associate responses (e.g., reply, comments) with the target of their response. Based on the premise that variations in the structural properties of the underlying rumour threads could play an important role in identifying and classifying the veracity of rumours tweets and their responses, the following network characteristics were used: DFS and BFS priority, degree centrality, betweenness centrality, closeness centrality, HITS hub score and PageRank.

Twitter metadata - Several Twitter metadata were retrieved and included in our model. The following list of information were considered to be relevant and were added as separate features: the number of characters in a tweet, the number of favorites and retweets and the number of days since initial posting. For each user, the information whether the user account is verified, the number of user's followers, the number of statuses, the number of friends, the number of favorites, number of times listed and how long the user has had a Twitter account were also included. Some additional information on the graphical design choices a user made were also considered. For example, whether there is a background image or a default image was used and the colors selected for the text, border, sidebar, background and hyperlinks.

Similarity measure - Transmission of context shared by the initial rumour tweet and its responses was postulated to be important for the task of classifying the support of the responses (task A). Consequently, it was decided to calculate the differences between the feature vectors of each response tweet and the rumour tweet it replies to.

3.4 Feature Selection

The feature extraction process resulted in a set of approximately 14000 features for each tweet, in addition to the ones resulting from textual analysis of the user profile. Appropriate feature selection is essential for achieving good performance and avoiding overfitting by removing uninformative, redundant or noisy information. First, the features were normalized to the interval [-1, 1] using generalized Min-Max normalization. Then, a Random Forest classifier was used as a basis for calculating the information gain of each feature. The features with an above-average information gain score were selected, reducing the number of features by a factor of 10. It was decided not to eliminate any correlated features as the number of samples is quite low and any detected correlation might be due to statistical falsehood.

3.5 Model training

At the outset of our explorations in order to circumvent the difficulties imposed by a multi-class classification problem, presented as task A, a hierarchical one-vs-rest approach was adopted. To this end, a binary classifier was used to classify each response tweet into two classes, comment or non-comment; the next step classifies each non-comment tweet as a query or a non-query type. In the end, each non-query response was classified as being in support or denial of the original rumourous post.

Balancing of the datasets was a necessary step during the training of the models for task A, as the distribution of the original four SDQC classes was not uniform. The balancing process was performed by repeated random sampling. Thirty candidate datasets with uniform class distribution were formed by random sampling the more prevalent class and the best candidate was chosen using 3-fold cross validation with a Naive Bayes classifier.

The models used for Task A and B are ensembles of six different classifiers, including: Naive-Bayes, K-Nearest-Neighbours, Logistic Regression, Support Vector Machine, Neural Network and Random Forest. The ensemble classifier for task A was operationalized as majority voting, while for task B, probability-weighted voting was used (probability weights correspond to the confidence level of the veracity scores).

In order to train and evaluate the ensemble, each dataset was split into a training and a validation set using 3-fold cross-validation with stratified sampling. This process was repeated 10 times and the final evaluation scores were calculated by averaging the scores from each iteration. For some training runs, the parameters of the classifiers had been optimized by 3-fold cross validation using grid search. However, no notable improvements in evaluation results were witnessed.

The system[1] was implemented using Python 3.6, with the ScikitLearn, NLTK, gensim and NetworkX packages Pedregosa et al. (2011); Bird et al. (2009); Řehůřek and Sojka (2010); Hagberg et al. (2008).

4 Discussion of results

The performance results obtained by the model that included all categories of features, discussed in the previous section, when tested on our validation set are presented in Table 1. The same model was used to create our final submission for SemEval Task 7. It is worth noting that the best results were obtained for identifying queries in task A, obtaining an accuracy of 0.784, demonstrating that the model accuracy rivals the performance of domain experts. We would like to highlight the consideration that no external sources were used for determining the veracity of a rumour posting, including the resources deemed as appropriate by the task organizers, which might explain the lower performances on the second task B.

Although the aforementioned results provide compelling evidence suggesting that fusing linguistic analysis, metadata, user profiles and rumour thread structure can lead to satisfactory results for classifying the stance of a response to a

[1]The complete source code is available at: `https://github.com/Bani57/rumourEval2019`.

Model	Accuracy	Precision	Recall	F1	ROC AUC	Log loss
Task A - Comment	0.767632	0.773194	0.767632	0.766449	0.767632	/
Task A - Query	0.784186	0.790382	0.784186	0.783002	0.784190	/
Task A - Support/Deny	0.682915	0.687301	0.682915	0.681009	0.682702	/
Task B - Veracity	0.681707	0.693103	0.681707	0.677606	0.681707	0.600458

Table 1: Evaluation metrics on a randomly-created validation set

rumourous tweet, we have conducted two ablation studies, to find evidence of the performance gains that could be contributed to each category of features on both tasks.

The present study suggests that text analysis of rumourous tweets is the most important, yet not the sole constituent element when detecting the veracity of rumourous tweet and distinguishing the stance of the social response. While we are not in position to ascribe sound theoretical reason to all effects, we present the trends that appeared interesting and highlight the sensitivity of the performance results towards a particular category of features.

Table 2 displays the F1 results of ablation study when training the models for task A, removing one category of features at a time. The results affirm that capturing the metadata relating to the users profile and structural properties of the tree-like threads of tweet exchange complement the linguistic features and improve the predictive accuracy, especially for distinguishing between query, support and denial stance in rumour replies.

The ablation analysis strongly demonstrates that Twitter metadata has the most dramatic effect on distinguishing **comments** from all other rumour responses - removal of this category results in lowering the F1 values by 0.17. It appears that user's historical and behavioral metadata add to the prediction performance complementing the relevant n-grams in the content of the tweet and user profile. Some of the most relevant indicators in the rumourous comment include: end of sentence punctuation, hashtags, user mentioned in the tweet, pronouns and words such as: *reported*, *happening*, as well as n-grams extracted from the user profile, such as: *blog*, *I am*, *concerned citizen*, *culture*, *enthusiast*, *living*, etc.

Language model and word vectors representation of the content of tweets and user profiles were indicated as better predictors when identifying replies in the form of **queries**, resulting in 0.12 decrease in F1 values, if removed from the model. Some of the most relevant features were not surprisingly related to detecting question forms: *why*, *where*, *question mark at the end of a sentence*, *who was*, *what is*, *confirm*, *need*, and a number of word2vec clusters. In addition, language style, sentiment and Twitter interaction threads have also ranked in the top 10% of the most relevant features.

Sentiment and structural features have a more notable effect on discriminating between **supportive and denying responses**. We could hypothesize that the affective content of a tweet is a crucial indicator when distinguishing positive (confirmative) vs. negative (opposing) opinion toward the source tweet with rumourous claim. The top 10% of the best predictive indicators were the sentiment words, and n-grams, such as: *not*, *believe*, *know*, *oh*, *ugh*, *such*, *yeah*, *understand*, socially offensive words etc.

Table 3 shows the cross-entropy loss yielded by the models after each ablation step trained for predicting the **veracity** of a rumour (task B). The findings highlight the ability of language indicators to model the truthfulness or deception of a claim. The features with most predictive power were language model and word vector clusters, especially numbers and URLs in the tweet text, and words obtained from the user profile, such as: *delivering you*, *insightful analysis*, *breaking news*, *contact*, *facebook*, *latest*, *tweets*, *we*, *views*, EMAIL, *online news*, *around the world*, *channel*, *bbc*, *bbcsport*, *cnn*, etc.

The current findings demonstrate surprisingly low accuracies, (F1 = 0.21645 for Task A, F1 = 0.3326 for Task B) when evaluated on the testing dataset, although in line with the results of previous tasks on the same dataset Derczynski et al. (2017). Collecting larger quantities of Twitter data and optimization techniques could improve the consistency of the results obtained on the training and validation sets. Importance of close inspection of data, and comparative analysis with other research on the same task could better support the

Ablation	Comment F1	Query F1	Support/Deny F1
Baseline	0.795715	0.804499	0.707801
Without language style	0.795715	0.794216	0.707801
Without n-grams and embeddings	0.785013	0.681081	0.618934
Without sentiment	0.784082	0.791145	0.666029
Without network structure	0.793228	0.789274	0.693583
Without Twitter metadata	0.628665	0.804499	0.691252

Table 2: Evaluation results from the ablation experiment for Task A performed on a randomly-created validation set

Ablation	Veracity cross-entropy loss
Baseline	0.591780
Without language style	0.605128
Without n-grams and embeddings	0.667380
Without sentiment	0.600889
Without network structure	0.591780
Without Twitter metadata	0.596146

Table 3: Evaluation results from the ablation experiment for Task B performed on a randomly-created validation set

interpretation of the results. We defer such discussion and directions for future research, until a detailed analysis of misclassified cases is done and proper treatment and improvements of such scenarios could be speculated.

5 Conclusion

The present research explores a hybrid approach to the problem of analyzing the veracity of rumours and the support for rumours on social media platforms. Following the results of previous research in this field, different combinations of features were examined, while also leveraging a variety of tangible indicators not accounted for in related research. The recurrent challenges in fully elucidating the language ambiguities of complex phenomena such as rumour spreading led us in a direction of including distal contextual indicators. In particular, the models were augmented with language indicators extrapolated from the content of user profiles, Twitter metadata, and thread structural characteristics of rumourous tweets. Their relevance and predictive effects have been confirmed with the results, providing exciting directions for further research on the problem.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Prashant Bordia and Nicholas Difonzo. 2004. Problem solving in social interactions on the internet: Rumor as social cognition. *Social Psychology Quarterly*, 67(1):33–49. https://doi.org/10.1177/019027250406700105.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. Semeval-2017 task 8: Rumoureval: Determining rumour veracity and support for rumours. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 69–76, Vancouver, Canada. Association for Computational Linguistics. http://www.aclweb.org/anthology/S17-2006.

Genevieve Gorrell, Kalina Bontcheva, Leon Derczynski, Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2019. SemEval-2019 Task 7: RumourEval: Determining rumour veracity and support for rumours. In *Proceedings of SemEval*. ACL. https://arxiv.org/pdf/1809.06683.pdf.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA.

Clayton J Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*.

Michal Lukasik, Trevor Cohn, and Kalina Bontcheva. 2015. Classifying tweet level judgements of rumours in social media. *Proceedings of the 2015 Conference on Empirical Methods in Natural Lan-*

guage Processing. http://dx.doi.org/10.18653/v1/D15-1311.

Tsvetomila Mihaylova, Preslav Nakov, Lluis Marquez, Alberto Barron-Cedeno, Mitra Mohtarami, Georgi Karadzhov, and James Glass. 2018. Fact checking in community forums. In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*. https://arxiv.org/pdf/1803.03178.pdf.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Warren A. Peterson and Noel P. Gist. 1951. Rumor and public opinion. *American Journal of Sociology*, 57(2):159–167. https://doi.org/10.1086/220916.

Vahed Qazvinian, Emily Rosengren, Dragomir R. Radev, and Qiaozhu Mei. 2011. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1589–1599, Stroudsburg, PA, USA. Association for Computational Linguistics. http://dl.acm.org/citation.cfm?id=2145432.2145602.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016a. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PLOS ONE*, 11(3):1–29. https://doi.org/10.1371/journal.pone.0150989.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016b. Pheme rumour scheme dataset: journalism use case. https://figshare.com/articles/PHEME_rumour_scheme_dataset_journalism_use_case/2068650/2.

BLCU_NLP at SemEval-2019 Task 7: An Inference Chain-based GPT Model for Rumour Evaluation

Ruoyao Yang, Wanying Xie, Chunhua Liu, Dong Yu (✉)
Beijing Language and Culture University, Beijing, China
{yangruoyao97, xiewanying07, chunhualiu596}@gmail.com
yudong@blcu.edu.cn

Abstract

Researchers have been paying increasing attention to rumour evaluation due to the rapid spread of unsubstantiated rumours on social media platforms, including SemEval 2019 task 7. However, labelled data for learning rumour veracity is scarce, and labels in rumour stance data are highly disproportionate, making it challenging for a model to perform supervised-learning adequately. We propose an inference chain-based system, which fully utilizes conversation structure-based knowledge in the limited data and expand the training data in minority categories to alleviate class imbalance. Our approach obtains 12.6% improvement upon the baseline system for subtask A, ranks 1st among 21 systems in subtask A, and ranks 4th among 12 systems in subtask B.

1 Introduction

With the universality of the Internet, social media has become the main channel for acquiring and exchanging information. However, the free flow of information has given rise to the prevalence of rumours, among which fake ones are harmful since they are generally convincing and hard to distinguish. To address this problem, we need automatic rumour veracity classification on social media. A large amount of rumour stance instances on social media have been employed to assist the model in making better predictions regards the rumour's veracity.

Rumour stance classification and rumour veracity classification are two subtasks of SemEval 2017 Task 8 (Derczynski et al., 2017) and SemEval 2019 Task 7 (Gorrell et al., 2018). Subtask A predicts the stance of a post replying to a rumourous post, in terms of supporting, denying, querying and commenting the rumour. Subtask B anticipates the veracity of a rumour as true or false given the rumourous post and a set of additional resources.

Apart from variations in models, research in this area mainly focuses on the special characteristics of data coming from social media: conversation structure, rich intrinsic features, skewed distribution toward the *comment* class in rumour stance data and scarcity of available data for rumour veracity classification. While most pioneering works treated rumour evaluation as a single-tweet task, attempts to utilize the conversation structure included pairing source and replies together to make up input (Singh et al., 2017), and adopting the full conversation thread as input in the time sequence (Kochkina et al., 2017). With the realization of rich features hidden in tweet contexts, Qazvinian et al. (2011) was one of the first who extracted them and combined them with model input. The feature sets were augmented during the following work. In trying to acquire more comprehensive information, not only features of the tweet for prediction were taken into consideration, but also features from its conversation context (Enayet and El-Beltagy, 2017). To address the class imbalance problem, Wang et al. (2017) transformed subtask A into a two-step classification task: they first classified comments and non-comments, and then categorized non-comments into the other three classes. Finally, in order to make up for the absence of abundant accessible training data in the rumour veracity classification task, external resources usage such as Wikipedia dumps and news articles was encouraged in both RumourEval contests.

In our work, we find that simply taking the whole conversation as input is inadequate. We have to recognize the role each part assumes in the conversation thread and mark them accordingly in the input. Instead of filtering features solely based on the system performance, we prefer to run a feature selection before adding to the system and choose those that can bring a high deviation degree between data categories. Following the feature extraction work of Enayet and El-Beltagy (2017), we consider introducing more features from the conversation context to further assist model judgment. We alleviate class imbalance in stance classification by expanding training data in the under-represented classes with pre-screened external data from similar datasets. At last, we approach the data insufficiency issue by setting an average length limit and cutting the overlength ones to enlarge training data.

We propose an inference chain-based system for this paper. A conversation thread starts with a source tweet, and follows by replies, in which each one responds to an earlier one in time sequence. When we infer the stance of one tweet, the source or earlier replies in the same thread can give abundant additional hints. Therefore, we take each conversation thread as an inference chain and concentrate on utilizing it to solve the data issues discussed earlier.

Our approach for both tasks is fine-tuned on Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), a model that has performed well in 9 NLP tasks. Our work primarily focuses on subtask A rumour stance classification, in which we expand training data from similar datasets, extract features and join separate parts to form input according to their roles in inference chain. For subtask B rumour veracity classification, we apply similar feature extraction and input concatenation process, except for replacing the data expansion step with data slicing. With the above implementation, our model outperforms all other systems in subtask A and places 4th in subtask B in SemEval 2019.

2 System Description

We propose a system that focuses on inference chain-based knowledge enhancement. The operations involved are displayed in Figure 1. We first perform data preprocessing on the raw dataset (Section 2.1) to fetch tweet content and facilitate the subsequent feature extraction step. Then we implement two data extension mechanisms on the training data: to relieve class imbalance toward one category in subtask A, we expand training data with external datasets (Section 2.2); to alleviate data sparsity and try to avoid under-fitting

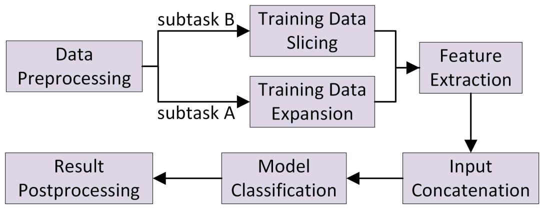


Figure 1: The operational flowchart for our system.

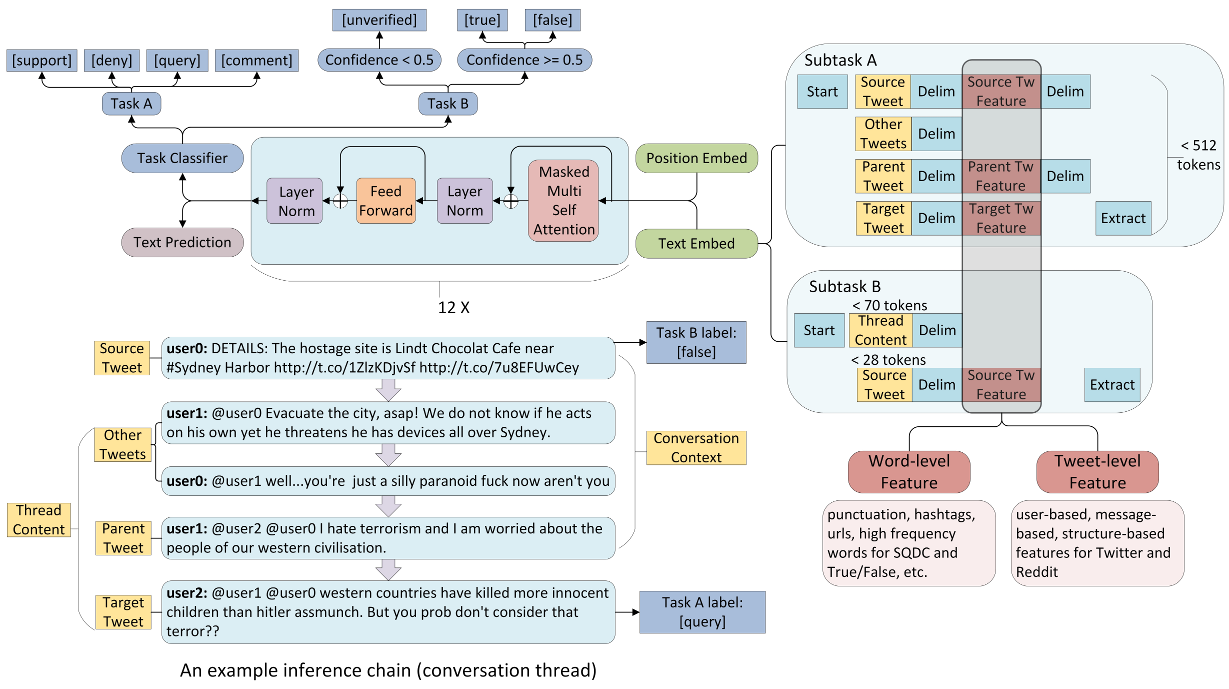
on the training set in subtask B, we set a length limit for each part of the input and split data instances accordingly (Section 2.3) to enlarge the amount of data. In trying to better utilize inference chain-based features, we extract word-level and tweet-level features and filter them for each subtask (Section 2.4). Finally, we concatenate each part in the inference chain with its features together (Section 2.5) and feed them in the model. After model classification, we adjust some of the results(Section 2.6) according to the organizers' requirements.

An illustration of our system is shown in Figure 2. How our base model GPT performs on the two classification tasks is depicted in the upper left corner. The right side presents how we organize our input for subtask A and B. In the lower left corner, we give an example of an inference chain (conversation thread) in the training data. To help understand, we define each part of it for the rest of this passage. For subtask A, we divide an inference chain into four parts: *source tweet*, *other tweets*, *parent tweet* and *target tweet*. For subtask B, an inference chain constitutes of a *source tweet* and a *thread content*. To better clarify, a *thread content* is defined as a whole inference chain excluding the source tweet, and we also define a whole inference chain excluding the target tweet as a *conversation context*. As depicted in the upper left corner, the goal is to predict the stance of a target tweet toward a rumour for subtask A, and the veracity of a rumour(usually contained in the source tweet) for subtask B.

2.1 Data Preprocessing

After extracting the tweet content out of the original data, we first perform word tokenization with Stanford CoreNLP[1], Spacy[2] and NLTK tools[3], among which the result of Stanford CoreNLP

[1]https://stanfordnlp.github.io/CoreNLP/
[2]https://spacy.io/
[3]http://www.nltk.org/



user0: DETAILS: The hostage site is Lindt Chocolat Cafe near #Sydney Harbor http://t.co/1ZlzKDjvSf http://t.co/7u8EFUwCey

user1: @user0 Evacuate the city, asap! We do not know if he acts on his own yet he threatens he has devices all over Sydney.

user0: @user1 well...you're just a silly paranoid fuck now aren't you

user1: @user2 @user0 I hate terrorism and I am worried about the people of our western civilisation.

user2: @user1 @user0 western countries have killed more innocent children than hitler assmunch. But you prob don't consider that terror??

Figure 2: An overview of our system. The upper left corner is a diagram of how our base model GPT performs on the two classification tasks. The upper right corner is how we organize our input for subtask A and B. The lower left corner is a data example which helps explain "inference chain", "source tweet", "other tweets", "parent tweet", "target tweet", "thread content" and "conversation context" in the input. The lower right corner is the features we combined in the input.

proves to be the best. Then we fix some tokenization inaccuracies in the result. Blanks, emoticons and uncommon punctuations are removed to reduce the amount of Out of Vocabulary (OOV). Besides, we transform all user mentions into "@" and all URLs into "urlurlurl" in order to increase the repetition rate of these features and allow the model to learn them better. Finally, we convert all letters into their lower case for the same purpose.

Training Set			
Class	**Origin**	**Extended**	**Sum**
support	925(18%)	797	1722(23%)
deny	378(7%)	696	1074(14%)
query	395(8%)	912	1307(17%)
comment	3519(67%)	0	3519(46%)

Table 1: Distribution of tweets between classes before and after data expansion in the training set for subtask A.

2.2 Traning Data Expansion

We find that the data distribution in the training set is skewed toward *comment* in subtask A, which explains why it is hard for the model to reach high precision and recall scores in the other three classes. Thus we expand the training data with datasets on similar tasks with labels corresponding to the three minority classes in seeking for better class balance and more sufficient training.

For *support* and *deny*, we take each claim as both the target tweet and its conversation context in stance classification datasets SemEval 2016 task 6 dataset (Mohammad et al., 2016), Emergent (Ferreira and Vlachos, 2016), and twitter sentiment analysis dataset sentiment140 (Go et al., 2009). For *query*, we use passages as the conversation context, unanswerable questions as the target tweet in reading comprehension datasets SQuAD 2.0 (Rajpurkar et al., 2018) and CoQA (Reddy et al., 2018).

The acquirement of extended data takes two steps. We first calculate the sentence similarity of each data piece in external datasets with all sentences in the original dataset in terms of Levenshtein Distance, and only keep instances whose minimal distances with the original dataset are below 0.7. Then we test them in a model trained with the original dataset. If the model prediction is identical to the label we expect, we append this instance in our training set for subtask A. The data distribution before and after data expansion is shown in Table 1.

2.3 Traning Data Slicing

The training set for subtask B only has 327 pieces. Since the organizers define subtask B as a binary classification task, which classifies instances into two categories: *true* and *false*, and takes pieces whose confidence scores below 0.5 as *unverified*, the *unverified* class in the training set is entirely useless, which takes up 98 pieces. Discard of this class only makes the data scarcity problem worse and may lead to under-fitting on the training set.

We try to extend the training set from a different angle. We look at the distribution of sequence length and set a length restriction for each conversation context to 70 tokens and each target tweet to 28 tokens. For those that exceed the limit in the training set, we truncate each piece to the edge to create multiple instances and thus enlarge the training set. For the development set and test set, the length restrictions are also set, but only the first truncated piece for each instance is taken as input. Although data slicing may hurt long-distance dependency, the experiment result shows that this method performs better than the original.

2.4 Feature Extraction

Inspired by the features extracted by Kochkina et al. (2017) and Bahuleyan and Vechtomova (2017), we collect 56 word-level features and 16 tweet-level features. For word-level features, we calculate their distribution percentage on the four categories in subtask A and the three categories in subtask B in the training and development set, and only apply those that mark a clear distinction between the classes for each subtask. For the numerical tweet-level features, we cluster each one into several groups according to their values and determine a common value for the whole group. Where to add the features is illustrated on the right side in Figure 2.

After selection, the word-level features we apply to subtask A and B are as below:

Subtask A: Whether the tweet content has question marks, hashtags, URLs, “RT”(refers to retweet), positive words, negative words, swear-words, query words, forbidding verbs, accusing verbs, complaining verbs, warning verbs, permitting verbs, praising verbs, etc.

Subtask B: Whether the tweet content has exclamation marks, positive words, negative words, query words, false synonyms, false antonyms, declaring verbs, confirming verbs, arguing verbs, etc.

The tweet-level features we add for subtask A are as below:

Features for both Twitter and Reddit: Tweet favourite count, tweet depth in the thread and whether the user has user description.

Twitter-specific features: User-related features include whether the user is verified, user-related URLs, whether the user uses the default profile, user followers count, user friends count, user listed groups count, and user statuses count. Tweet-related features include tweet retweet count.

Reddit-specific features: Whether the tweet has self-text, tweet kind, whether the tweet is archived, whether the tweet is edited, whether the user is a submitter.

We’ve tried to add the above tweet-level features to input for subtask B, but unfortunately observe no performance improvement in our experiment, so the official submission for subtask B involves no tweet-level features.

As indicated in Enayet and El-Beltagy (2017), whether the users of the source and parent tweet are verified are useful in performance improvement for subtask A. So we speculate that word-level features and other tweet-level features may also be necessary hints for model prediction. We find that the model always creates mispredictions because the stance of the target tweet toward the rumour is indirect. Since many tweets express direct stances toward their parent tweets, we can infer their stances toward the rumour with the help of their parent. In addition, the model often mistakes the stance of target tweet toward source tweet as its stance toward the rumour. But in cases where the source tweet expresses *deny* towards the rumour, the two stances above are not consistent. Therefore, parent tweet and source tweet assume significant roles in an inference chain and we need to acquire more knowledge from them to improve the prediction accuracy of the target tweet. So we apply all filtered word-level and tweet-level features for the source, the parent and the target to the input for subtask A.

2.5 Input Concatenation

As described above, the model is likely to make correct predictions when it learns information from earlier tweets in the inference chain. We plan to concatenate the contexts in the inference chain with the target tweet to make up for input. The input concatenation method is depicted in the upper right corner of Figure 2. We concatenate *source tweet* with *thread content* to form an input structure for subtask B. We apply two methods in subtask A. At first we concatenate the *conversation context* and the *target tweet* with the same structure as subtask B. However, this method often results in the model's confusion between the stance of the target tweet toward the source and the parent. Thus, we decide to mark the position of the source and the parent in the conversation thread with delimiters and connect them with the rest of the conversation (other tweets) together. Each part is marked by delimiters and features are inserted behind them in the input.

2.6 Model

We employ OpenAI GPT that has been pretrained on BooksCorpus (Zhu et al., 2015) as our base model for task-specific fine-tuning. We've tried different activation functions, optimizers and hyperparameters but observe no performance improvement. So we use the default model configuration for the official submission. GPT requires its input length to be less than 512 tokens. For inputs that exceed this limit, we choose to cut off the "other tweets" sequence to fit this restriction.

Result Postprocessing As required by the organizers, we transform the labels of all blank instances into *comment* for subtask A, and take pieces whose confidence levels are below 0.5 as *unverified* in subtask B in the model prediction results.

3 Evaluation

We conduct experiments on data expansion, input format adjustment and word & tweet-level feature adding for subtask A, and perform data slicing and feature adding for subtask B. The dataset we used for this task is obtained from Zubiaga et al. (2016). We primarily focus on achieving a higher macro f1 score for both subtasks. We also display the comparison of the results on the test set between our official system and the baseline systems provided by the organizer.

3.1 Experiment Results

The following are the steps we conduct experiments toward creating the system for official submission.

Subtask A Our experiments conduct on the development set for subtask A is shown in Table 2. With the first input format (A), we achieve an initial result of 53.48%. Combining word-level, tweet-level features (B), and processing data expansion (C) brings an increase of 0.99% and 1.61% respectively. After converting to the second input format (D), we've seen a rise of 2.03% compared to A. Training data expansion (E) is also implemented on this input, but a decrease of 0.86% is observed. We suspect that after enhancing the percentage of the source and parent tweet in new input, data dissimilarity brought by the external dataset is aggravated. However, this inferior position is reversed by employing features. Word-level features (F) alone bring a 2.04% growth. Though tweet-level features alone haven't led in any extra increase, adding them together with word-level features (G) produces a result above 56%. Our final result for subtask A is ensembled on three runs (F1, F2, G) that achieve the best performance on the development set.

Subtask A		
	Systems	**MacroF**
A.	GPT(1st input)	0.5348
B.	GPT(1st input)+WF+TF	0.5447
C.	GPT(1st input)+DE	0.5509
D.	GPT(2nd input)	0.5551
E.	GPT(2nd input)+DE	0.5465
F1.	GPT(2nd input)+DE+WF	**0.5644**
F2.	(another run)	**0.5669**
G.	GPT(2nd input)+DE+WF+TF	**0.5631**

Table 2: Ablation results on the development set for subtask A. MacroF means macro f1 score. 1st input consists of conversation context and target tweet, while a 2nd input constitutes of source tweet, other tweets, parent tweet and target tweet. WF and TF refer to word-level and tweet-level features respectively. DE represents data expansion.

Subtask B Our experiments conducted on the development set for subtask B is shown in Table 3. Word features bring an increase of 3.71%. Tweet features prove to be disruptive and lead to a drop of up to 11.6%. So we discard this type of feature in subtask B entirely. The result we've submitted for subtask B is ensembled on two runs (B) that get the highest values on the development set.

Final Result Our final result on the test set and the comparison with the baseline systems are shown in Table 4. BranchLSTM and NileTMRG are two baselines implemented by the organizers.

Subtask B		
	Systems	**MacroF**
A.	GPT+DS	0.4701
B.	GPT+DS+WF	**0.5072**
C.	GPT+DS+WF+TF	0.4143

Table 3: Ablation results on the development set for subtask B. DS refers to training data slicing. WF and TF are the same meaning as Table 2.

Our system is 12.6% higher than the baseline system in macro f1 for subtask A, but 8.3% and 5.6% lower in macro f1 for subtask B compared with BranchLSTM and NileTMRG respectively. Our system ranks first in subtask A and fourth in subtask B.

Official Submission			
	Subtask A	Subtask B	
System	**MacroF**	**MacroF**	**RMSE**
BranchLSTM	0.493	**0.336**	0.781
NileTMRG		0.309	**0.769**
Our System	**0.6187**	0.2525	0.8179

Table 4: Official submission results on the test set for our system and the organizers' baselines.

3.2 Error Analysis

We perform error analysis for the results on the test set. Classification reports for both subtasks are provided in Table 5. The problem for subtask A lies in that the model often confused the *comment* class with the other three classes. The possible reason can be the relatively larger proportion of *comment* data. None of the classes performs well in prediction for subtask B, though results are better for the *true* class comparing with the other two classes. Precision, recall and macro f1 scores go extremely low for the *unverified* class. A reasonable explanation may be the *unverified* class is not directly acquired from the model but comes from the other two classes.

3.3 Comparison with ESIM

Since rumour evaluation can be seen as a task that given a target tweet and its background conversation, infer the target label of this tweet, they can be treated as a subtask of natural language inference, so we employ ESIM (Chen et al., 2016) in this task.

The results are shown in Table 6. The input format is the same as subtask B in GPT. We also try to apply features and change word embeddings in ESIM. But no results reach the result in GPT with an equivalent configuration. So the base model we employ for the system is GPT instead of ESIM.

Subtask A				
	prec.	**rec.**	**f1**	**distribution**
support	0.89	0.93	0.91	1476
deny	0.45	0.51	0.48	101
query	0.62	0.59	0.60	93
comment	0.66	0.38	0.48	157
Subtask B				
	prec.	**rec.**	**f1**	**distribution**
true	0.46	0.47	0.47	40
false	0.22	0.13	0.16	31
unverified	0.09	0.20	0.13	10

Table 5: Classification report on the test set for both subtasks.

Subtask A		
	Systems	**MacroF**
A.	ESIM(glove)	0.434
B.	ESIM(glove)+WF+TF	0.452
C.	ESIM(google news)+WF+TF	0.466
Subtask B		
	Systems	**MacroF**
A.	ESIM(glove)	0.473

Table 6: ESIM results on the development set for both subtasks.

4 Conclusions

We introduce a framework with a strong focus on inference chain-based knowledge enhancement for determining rumour stance and veracity in SemEval 2019 task 7. In order to address the problems of class imbalance, training data scarcity, model's insufficient learning of features and tree-structured conversations, we employ data expansion, data slicing, feature extraction, and input concatenation mechanisms in our system respectively. Our system takes first place in subtask A and fourth place in subtask B.

In future, we would like to introduce synonyms, tweet similarity and sentiment features in our model to further facilitate the recognition of relations between tweets. We will also utilize the prediction results from stance classification, expand training data with external datasets and introduce additional knowledge base such as Wikipedia to assist the model prediction in rumour veracity classification.

Acknowledgments

This work is funded by Beijing Advanced Innovation for Language Resources of BLCU (TYR17001J); the Fundamental Research Funds for the Central Universities in BLCU (No.17PT05) and the BLCU Academic Talents Support Program for the Young and Middle-Aged.

References

Hareesh Bahuleyan and Olga Vechtomova. 2017. Uwaterloo at semeval-2017 task 8: Detecting stance towards rumours with topic independent features. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 461–464.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2016. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. Semeval-2017 task 8: Rumoureval: Determining rumour veracity and support for rumours. *arXiv preprint arXiv:1704.05972*.

Omar Enayet and Samhaa R El-Beltagy. 2017. Niletmrg at semeval-2017 task 8: Determining rumour and veracity support for rumours on twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 470–474.

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 1163–1168.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12).

Genevieve Gorrell, Kalina Bontcheva, Leon Derczynski, Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2018. Rumoureval 2019: Determining rumour veracity and support for rumours. *arXiv preprint arXiv:1809.06683*.

Elena Kochkina, Maria Liakata, and Isabelle Augenstein. 2017. Turing at semeval-2017 task 8: Sequential approach to rumour stance classification with branch-lstm. *arXiv preprint arXiv:1704.07221*.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41.

Vahed Qazvinian, Emily Rosengren, Dragomir R Radev, and Qiaozhu Mei. 2011. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1589–1599. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2018. Coqa: A conversational question answering challenge. *arXiv preprint arXiv:1808.07042*.

Vikram Singh, Sunny Narayan, Md Shad Akhtar, Asif Ekbal, and Pushpak Bhattacharyya. 2017. Iitp at semeval-2017 task 8: A supervised approach for rumour evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 497–501.

Feixiang Wang, Man Lan, and Yuanbin Wu. 2017. Ecnu at semeval-2017 task 8: Rumour evaluation using effective features and supervised ensemble models. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 491–496.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PloS one*, 11(3):e0150989.

BUT-FIT at SemEval-2019 Task 7: Determining the Rumour Stance with Pre-Trained Deep Bidirectional Transformers

Martin Fajcik, Lukas Burget, Pavel Smrz
Brno University of Technology, Faculty of Information Technology
612 66 Brno, Czech Republic
{ifajcik,burget,smrz}@fit.vutbr.cz

Abstract

This paper describes our system submitted to SemEval 2019 Task 7: RumourEval 2019: Determining Rumour Veracity and Support for Rumours, Subtask A (Gorrell et al., 2019). The challenge focused on classifying whether posts from Twitter and Reddit *support, deny, query,* or *comment* a hidden rumour, truthfulness of which is the topic of an underlying discussion thread. We formulate the problem as a stance classification, determining the rumour stance of a post with respect to the previous thread post and the source thread post. The recent BERT architecture was employed to build an end-to-end system which has reached the F1 score of 61.67 % on the provided test data. Without any hand-crafted feature, the system finished at the 2nd place in the competition, only 0.2 % behind the winner.

1 Introduction

Fighting false rumours at the internet is a tedious task. Sometimes, even understanding what an actual rumour is about may prove challenging. And only then one can actually judge its veracity with an appropriate evidence. The works of Ferreira and Vlachos (2016) and Enayet and El-Beltagy (2017) focused on predictions of rumour veracity in thread discussions. These works indicated that the veracity is correlated with discussion participants' stances towards the rumour. Following this, the SubTask A SemEval-2019 Task 7 consisted in classifying whether the stance of each post in a given Twitter or Reddit thread *supports*, *denies*, *queries* or *comments* a hidden rumour.

Potential applications of such a function are wide, ranging from an analysis of popular events (political discussions, academy awards, etc.) to quickly disproving fake news during disasters.

Stance classification (SC), in its traditional form, is concerned with determining the attitude of a source text towards a target text (Mohammad et al., 2016). It has been studied thoroughly for discussion threads (Walker et al., 2012; Hasan and Ng, 2013; Chuang and Hsieh, 2015). However, the objective of SubTask A SemEval-2019 Task 7 is to determine the stance to a hidden rumour which is not explicitly given (it can be often inferred from the source post of the discussion – the root of the tree-shaped discussion thread – as demonstrated in Figure 1). The competitors were asked to classify the stance of the source post itself too.

.@AP I demand you retract the lie that
people in #Ferguson were shouting "kill the police",
local reporting has refuted your ugly racism

Figure 1: An example of discussion's source post denying the actual rumour which is present in the source post – annotated with red cursive

The provided dataset was collected from Twitter and Reddit tree-shaped discussions. Stance labels were obtained via crowdsourcing. The discussions deal with 9 recently popular topics – Sydney siege, Germanwings crash etc.

The approach followed in our work builds on recent advances in language representation models. We fine-tune a pre-trained end-to-end BERT (Bidirectional Encoder Representations from Transformers) model (Devlin et al., 2018), while using discussion's source post, target's previous post and the target post itself as inputs to determine the rumour stance of the target post. Our implementation is available online.[1]

2 Related Work

Previous SemEval competitions: In recent years, there were two SemEval competitions targeting the stance classification. The first one focused on the setting in which the actual rumour was provided (Mohammad et al., 2016). Organizers of

[1]www.github.com/MFajcik/RumourEval2019

SemEval-2016 Task 6 prepared a benchmarking system based on SVM using hand-made features and word embeddings from their previous system for sentiment analysis (Mohammad et al., 2013), outperforming all the challenge participants.

The second competition was the previous RumourEval won by a system based on word vectors, handcrafted features[2] and an LSTM (Hochreiter and Schmidhuber, 1997) summarizing information of the discussion's branches (Kochkina et al., 2017). Other submissions were either based on similar handcrafted features (Singh et al., 2017; Wang et al., 2017; Enayet and El-Beltagy, 2017), features based on sets of words for determining language cues such as Belief or Denial (Bahuleyan and Vechtomova, 2017), post-processing via rule-based heuristics after the feature-based classification (Srivastava et al., 2017), Convolutional Neural Networks (CNNs) with rules (Lozano et al., 2017), or CNNs that jointly learnt word embeddings (Chen et al., 2017).

End-to-end approaches: Augenstein et al. (2016) encode the target text by means of a bidirectional LSTM (BiLSTM), conditioned on the source text. The paper empirically shows that the conditioning on the source text really matters. Du et al. (2017) propose target augmented embeddings – embeddings concatenated with an average of source text embeddings – and apply them to compute an attention based on the weighted sum of target embeddings, previously transformed via a BiLSTM. Mohtarami et al. (2018) propose an architecture that encodes the source and the target text via an LSTM and a CNN separately and then uses a memory network together with a similarity matrix to capture the similarity between the source and the target text, and infers a fixed-size vector suitable for the stance prediction.

3 BUT-FIT's System Description

3.1 Pre-processing

We replace URLs and mentions with special tokens $\$URL\$$ and $\$mention\$$ using tweet-processor[3]. We use spaCy[4] to split each post into sentences and add the $[EOS]$ token to indicate termination of each sentence. We employ the tokenizer that comes with the Hugging Face PyTorch re-implementation of BERT[5]. The tokenizer lowercases the input and applies the WordPiece encoding (Wu et al., 2016) to split input words into most frequent n-grams present in the pre-training corpus, effectively representing text at the subword level while keeping a 30,000-token vocabulary only.

	S	D	Q	C	Total
train	925	378	395	3519	5217
in %	18	7	8	67	
dev	102	82	120	1181	1485
in %	7	6	8	80	
test	157	101	93	1476	1827
in %	9	6	5	81	

Table 1: Distribution of examples across classes in the training/development/test data set. The examples belong to 327/38/81 training/development/test tree-structured discussions.

3.2 Model

Following the recent trend in transfer learning from language models (LM), we employ the pre-trained BERT model. The model is first trained on the concatenation of BooksCorpus (800M words) (Zhu et al., 2015) and English Wikipedia (2,500M words) using the multi-task objective consisting of LM and machine comprehension (MC) sub-objectives. The LM objective aims at predicting the identity of 15% randomly masked tokens present in the input[6]. Given two sentences from the corpus, the MC objective is to classify whether the second sentence follows the first sentence in the corpus. The sentence is replaced randomly in half of the cases. During pre-training, the input consists of two documents, each represented by a sequence of tokens divided by the special $[SEP]$ token and preceded by the $[CLS]$ token used by the MC objective, i. e., $[CLS]document_1[SEP]document_2[SEP]$. Input tokens are represented by jointly learned token embeddings E_t, segment embeddings E_s, capturing whether the word belongs into $document_1$ or $document_2$, and positional embeddings E_p.

[2]The features included: a flag indicating whether a tweet is a source tweet of a conversation, the length of the tweet, an indicator of the presence of URLs and images, punctuation, the cosine distance to the source tweet and all other tweets in the conversation, the count of negation and swear words, and an average of word vectors corresponding to the tweet.

[3]https://github.com/s/preprocessor

[4]https://spacy.io/

[5]https://github.com/huggingface/pytorch-pretrained-BERT

[6]The explanation of token masking is simplified; details can be found in the original paper (Devlin et al., 2018).

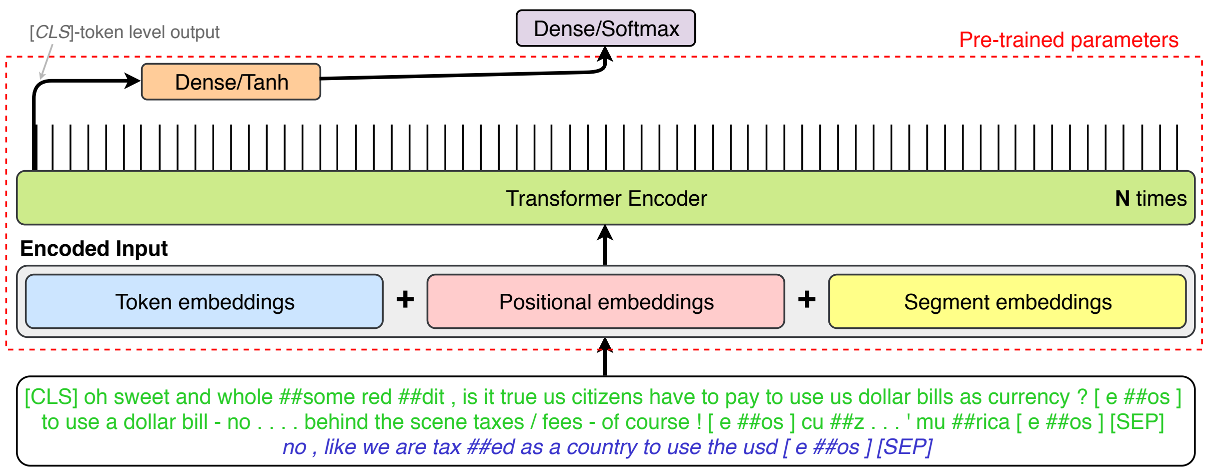

[CLS]-token level output
Dense/Softmax
Pre-trained parameters
Dense/Tanh
Transformer Encoder
N times
Encoded Input
Token embeddings
+
Positional embeddings
+
Segment embeddings
[CLS] oh sweet and whole ##some red ##dit , is it true us citizens have to pay to use us dollar bills as currency ? [e ##os] to use a dollar bill - no . . . . behind the scene taxes / fees - of course ! [e ##os] cu ##z . . . ' mu ##rica [e ##os] [SEP]
no , like we are tax ##ed as a country to use the usd [e ##os] [SEP]

Figure 2: An architecture of BUT-FIT's system. The text segment containing $document_1$ is green, the segment containing $document_2$ (the target post) is blue. The input representation is obtained by summing input embedding matrices $E = E_t + E_s + E_p \in \mathbb{R}^{L\times d}$, L being the input length and d the input dimensionality. The input is passed N times via the transformer encoder. Finally, the $[CLS]$ token-level output is fed through two dense layers yielding the class prediction.

Our system follows the assumption that the stance of discussion's post depends only on itself, on the source thread post and on the previous thread post. Since the original input is composed of two documents, we experimented with various ways of encoding the input (see Section 5), ending up with just a concatenation of the source and the previous post as $document_1$ (left empty in case of the source post being the target post) and the target post as $document_2$. The discriminative fine-tuning of BERT is done using the $[CLS]$ token level output and passing it through two dense layers yielding posterior probabilities as depicted in Figure 2. A weighted cross-entropy loss is used to ensure a flat prior over the classes.

3.3 Ensembling

Before submission, we trained 100 models differing just by their learning rates. We experimented with 4 different fusion mechanisms in order to increase the F1 measure and compensate for overfitting:

The `TOP−N` fusion chooses 1 model randomly and adds it to the ensemble. Then, it randomly shuffles the rest of the models and tries to add them into the ensemble one at the time, while iteratively calculating ensemble's F1 by averaging the output probabilities, effectively approximating the Bayesian model averaging. If a model increases the total F1 score, the model is permanently added to the ensemble. The process is repeated until no further model improving the ensemble's F1 score can be found. This procedure resulted in a set of 17 best models.

The `EXC−N` fusion chooses all models into the ensemble and then iteratively drops one model at the time, starting from that which dropping results in the largest increase of the ensemble's F1. The process stops when dropping any other model cannot increase the F1 score. Using this approach, we ended up using 94 models.

The `TOP−N`$_s$ is analogous to the `TOP−N` fusion, but we average pre-softmax scores instead of output class probabilities.

The `OPT−F1` fusion aims at learning weights summing up to 1 for the weighted average of output probabilities from models selected via the procedure used in the `TOP−N` strategy. The weights are estimated using modified Powell's method from the SciPy package to maximize the F1 score on the development dataset.

4 Experimental Setup

We implemented our models in PyTorch, taking advantage of the Hugging Face re-implementation (see Footnote 5), with the *"BERT-large-uncased"* setting, pre-trained using 24 transformer layers, having the hidden unit size of $d = 1024$, 16 attention heads, and $335M$ parameters. When building the ensemble, we picked learning rates from the interval $[1e{-}6, 2e{-}6]$. Each epoch iterates over the dataset in an ordered manner, starting by the shortest sequence. We truncate sequences at maximum length $l = 200$ with a heuristic – firstly we truncate the $document_1$ to length $l/2$, if that is not enough, then we truncate the $document_2$ to

	#Θ	Acc_{test}	macro F1_{dev}	macro F1_{test}	F1_S	F1_Q	F1_D	F1_C
Branch-LSTM	453K	84.10	-	49.30	43.80	55.00	7.10	91.30
FeaturesNN	205K	82.84	$45.46 \pm 1e{-2}$	$44.55 \pm 2e{-2}$	40.29	40.12	17.69	80.43
BiLSTM+SelfAtt	28M	83.59	$47.55 \pm 6e{-3}$	$46.81 \pm 6e{-3}$	42.21	45.20	17.75	81.92
BERT_{base}	109M	84.67	$51.40 \pm 1e{-2}$	$53.39 \pm 3e{-2}$	43.49	59.88	18.42	90.36
$\text{BERT}_{big-noprev}$	335M	84.33	$52.61 \pm 2e{-2}$	$52.91 \pm 4e{-2}$	42.37	55.17	24.44	90.15
$\text{BERT}_{big-nosrc}$	335M	84.51	$53.72 \pm 2e{-2}$	$55.13 \pm 3e{-3}$	43.02	56.93	26.53	90.51
BERT_{big}	335M	84.08	$56.24 \pm 9e{-3}$	$56.70 \pm 3e{-2}$	44.29	57.07	35.02	90.41
BERT_{big} EXC-N*	-	85.50	58.63	60.28	48.89	62.80	37.50	91.94
BERT_{big} TOP-N*	-	85.22	62.58	60.67	48.25	62.86	39.74	91.83
BERT_{big} OPT-F1	-	85.39	62.68	61.27	48.03	62.26	42.77	92.01
BERT_{big} TOP-N_s	-	85.50	61.73	**61.67**	49.11	64.45	41.29	91.84

Table 2: Overview of the results. The values for each single model were obtained by averaging results of 11 models. We report the mean and the standard deviation in these cases. #Θ denotes the number of parameters. Columns F1_S to F1_C report individual F1 scores for each class. All ensemble models have the F1 score optimized on the development dataset. BiLSTM+SelfAtt contains 4.2M parameters, without pre-trained BERT embeddings. $\text{BERT}_{big-nosrc}$ and $\text{BERT}_{big-noprev}$ denote system instantiations with an empty source and an empty target post, respectively. Note that the accuracy is biased towards different training data priors as shown in Table 1. SemEval submissions are denoted by *.

the same size. We keep the batch size of 32 examples and keep other hyperparameters the same as in the BERT paper. We use the same Adam optimizer with the L2 weight decay of 0.01 and no warmup. We trained the model on the GeForce RTX 2080 Ti GPU.

5 Results and Discussion

We compare the developed system to three baselines. The first one is the branch-LSTM baseline provided by the task organizers[7] – inspired by the winning system of RumourEval 2017. The second baseline (FeaturesNN) is our reimplementation of the first baseline in PyTorch without the LSTM – posts are classified by means of a 2-layer network (ReLU/Softmax), using only the features defined in Footnote 2. In the third case (BiLSTM+SelfAtt), we use the same input representation as in our submitted model but replace the BERT by an 1-layer BiLSTM network followed by a self-attention and a softmax layer, inspired by Lin et al. (2017).

The results are shown in Table 2. BERT models had to cope with a high variance during the training. This might be caused by the problem difficulty, the relatively small number of training examples, or the complexity of the models. To deal with the problem, we decided to discard all models with F1 scores of less than 55 on the development dataset and we averaged the output class probability distributions when ensembling. Our initial experiments used sequences up to the length of 512, but we found no difference when truncating them down to 200.

What features were not helpful: We tried adding a number of other features, including those indicating positive, neutral, or negative sentiment, and all the features used by the FeaturesNN baseline. We also tried adding jointly learned POS, NER, and dependency tag embeddings, as well as the third segment embeddings[8]. We also experimented with an explicit $[SEP]$ token to separate the source and the previous post in the BERT input. However, none of the mentioned changes led to a statistically significant improvement.

6 Conclusions and Future Directions

The system presented in this paper achieved the macro F1 score of 61.67, improving the baseline by 12.37%, while using only the source post of discussion, the previous post and the target post to classify the target post's stance to a rumour.

A detailed analysis of the provided data shows that the employed information sources are not sufficient to correctly classify some examples. Our future work will focus on extending the system by a relevance scoring component. To preserve the context, it will evaluate all posts in a given discussion thread and pick up the most relevant ones according to defined criteria.

[7] http://tinyurl.com/y4p5ygn7

[8] We tried adding the learned representations to the input the same way the segment/positional embeddings are added.

Acknowledgments

This work was supported by the Czech Ministry of Education, Youth and Sports, subprogram INTER-COST, project code: LTC18054.

References

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. *arXiv preprint arXiv:1606.05464*.

Hareesh Bahuleyan and Olga Vechtomova. 2017. Uwaterloo at semeval-2017 task 8: Detecting stance towards rumours with topic independent features. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 461–464.

Yi-Chin Chen, Zhao-Yang Liu, and Hung-Yu Kao. 2017. Ikm at semeval-2017 task 8: Convolutional neural networks for stance detection and rumor verification. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 465–469.

Ju-han Chuang and Shukai Hsieh. 2015. Stance classification on ptt comments. In *29th Pacific Asia Conference on Language, Information and Computation Proceedings of PACLIC 2015: Poster Papers*, page 27.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jiachen Du, Ruifeng Xu, Yulan He, and Lin Gui. 2017. Stance classification with target-specific neural attention networks. International Joint Conferences on Artificial Intelligence.

Omar Enayet and Samhaa R El-Beltagy. 2017. Niletmrg at semeval-2017 task 8: Determining rumour and veracity support for rumours on twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 470–474.

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 1163–1168.

Genevieve Gorrell, Kalina Bontcheva, Leon Derczynski, Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2019. SemEval-2019 Task 7: RumourEval: Determining rumour veracity and support for rumours. In *Proceedings of SemEval*. ACL.

Kazi Saidul Hasan and Vincent Ng. 2013. Stance classification of ideological debates: Data, models, features, and constraints. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1348–1356.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Elena Kochkina, Maria Liakata, and Isabelle Augenstein. 2017. Turing at semeval-2017 task 8: Sequential approach to rumour stance classification with branch-lstm. *arXiv preprint arXiv:1704.07221*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Marianela García Lozano, Hanna Lilja, Edward Tjörnhammar, and Maja Karasalo. 2017. Mama edha at semeval-2017 task 8: Stance classification with cnn and rules. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 481–485.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of the seventh international workshop on Semantic Evaluation Exercises (SemEval-2013)*, Atlanta, Georgia, USA.

Mitra Mohtarami, Ramy Baly, James Glass, Preslav Nakov, Lluís Màrquez, and Alessandro Moschitti. 2018. Automatic stance detection using end-to-end memory networks. *arXiv preprint arXiv:1804.07581*.

Vikram Singh, Sunny Narayan, Md Shad Akhtar, Asif Ekbal, and Pushpak Bhattacharyya. 2017. Iitp at semeval-2017 task 8: A supervised approach for rumour evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 497–501.

Ankit Srivastava, Georg Rehm, and Julian Moreno Schneider. 2017. Dfki-dkt at semeval-2017 task 8: Rumour detection and classification using cascading heuristics. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 486–490.

Marilyn A Walker, Pranav Anand, Robert Abbott, and Ricky Grant. 2012. Stance classification using dialogic properties of persuasion. In *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 592–596. Association for Computational Linguistics.

Feixiang Wang, Man Lan, and Yuanbin Wu. 2017. Ecnu at semeval-2017 task 8: Rumour evaluation using effective features and supervised ensemble models. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 491–496.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

A Supplemental Material

A.1 Dataset Insights

The dataset contains a whole tree structure and metadata for each discussion from Twitter and Reddit. The nature of the data differs across the sources (for example, the Reddit subset includes upvotes).

When analysing the data, we spotted several anomalies:

- 12 data points do not contain any text. According to the task organizers, they were deleted by users at the time of data download and been left in the data not to break the conversational structure.

- The query stance of some examples taken from subreddit DebunkThis[9] is dependent on the domain knowledge. The class of some examples is ambiguous; they should be probably labelled by multiple classes.

A.1.1 Domain knowledge dependency

Examples from subreddit DebunkThis have all the same format ”Debunk this: [statement]”, e.g. *”Debunk this: Nicotine isn't really bad for you, and it's the other substances that makes tobacco so harmful.”*. All these examples are labelled as queries.

[9]https://www.reddit.com/r/DebunkThis/

A.1.2 Class ambiguity

The source/previous post *”This is crazy! #CapeTown #capestorm #weatherforecast https://t.co/3bcKOKrCJB”* and the target post *”@RyGuySA Oh my gosh! Is that not a tornado?! Cause wow, It almost looks like one!”*, labelled as a comment in the dataset, might be seen as a query as well.

A.2 Additional Introspection

Figures 3, 4, 5, and 6 demonstrate attention matrices A, derived from the multi-head attention defined as:

$$A = \frac{QK^\top}{\sqrt{d_k}}, \qquad (1)$$

where $Q, K \in \mathbb{R}^{L \times d_k}$ are the matrices containing query/value vectors and d_k is the key/value dimension. The insights are selected from the heads at the first layer of the transformer encoder.

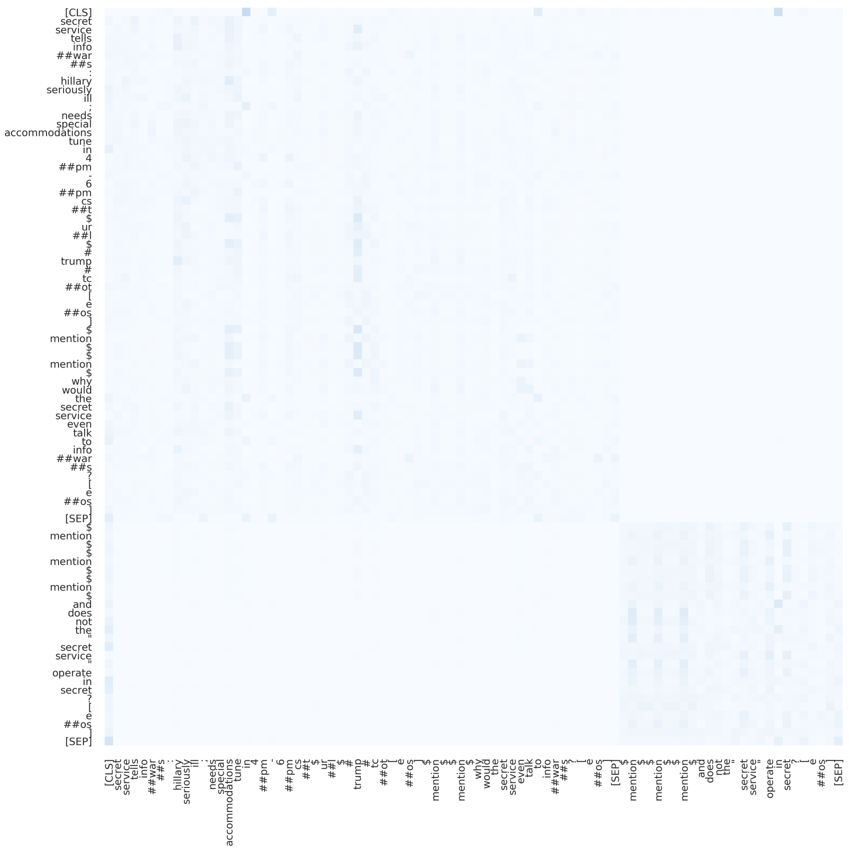


Figure 3: Intra-segment attention – the attention is made only between the subword units from the same segment.

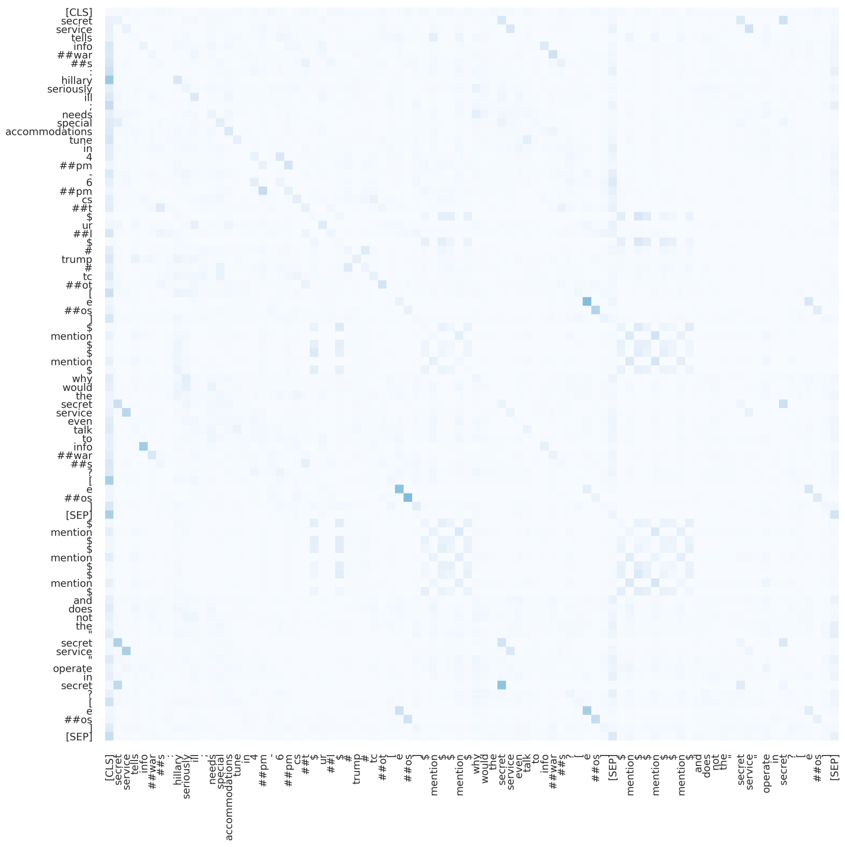


Figure 4: Attention matrix capturing the subword similarity.

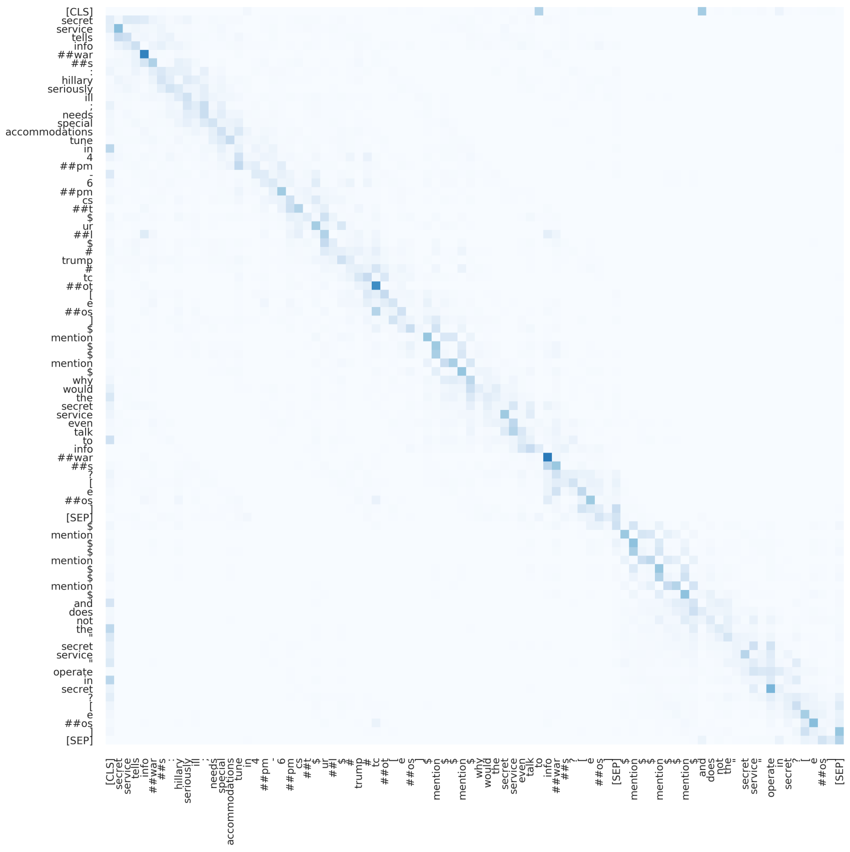


Figure 5: ’Soft’ local context aggregation.

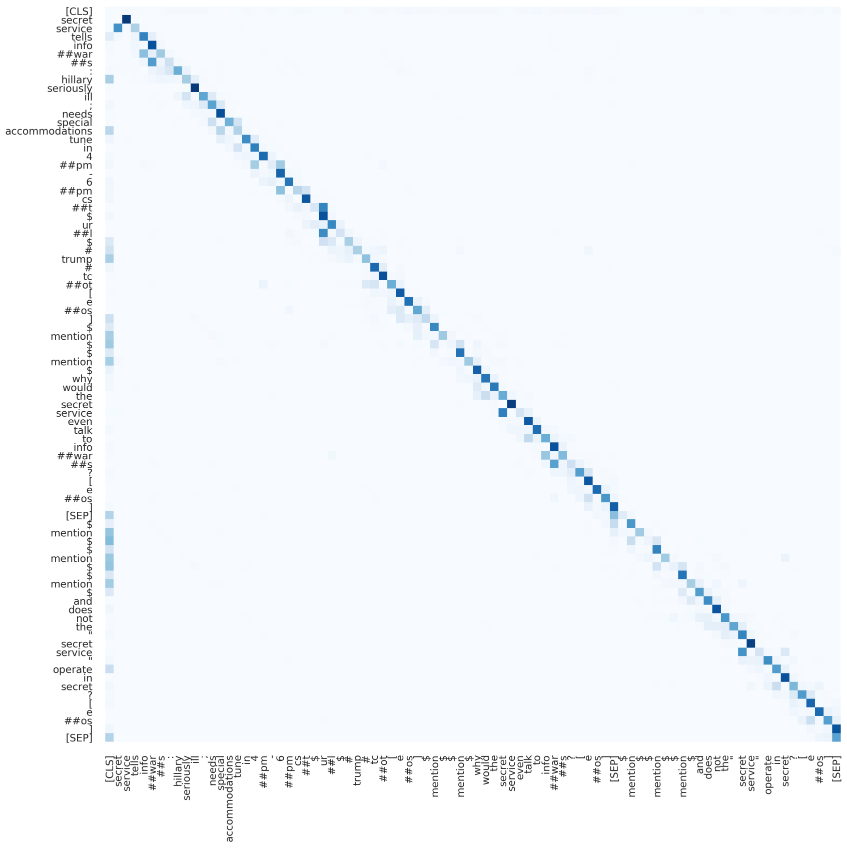


Figure 6: ’Hard’ local context aggregation – the signal is mostly sent further to another transformer encoder layer.

CLEARumor at SemEval-2019 Task 7: ConvoLving ELMo Against Rumors

Ipek Baris[1,*] and **Lukas Schmelzeisen**[1,*] and **Steffen Staab**[1,2]
[1]Institute for Web Science and Technologies (WeST), University of Koblenz-Landau, Germany
[2]Web and Internet Science Group (WAIS), University of Southampton, United Kingdom
ibaris@uni-koblenz.de, lukas@uni-koblenz.de, staab@uni-koblenz.de

Abstract

This paper describes our submission to SemEval-2019 Task 7: RumourEval: Determining Rumor Veracity and Support for Rumors. We participated in both subtasks. The goal of subtask A is to classify the type of interaction between a rumorous social media post and a reply post as support, query, deny, or comment. The goal of subtask B is to predict the veracity of a given rumor. For subtask A, we implement a CNN-based neural architecture using ELMo embeddings of post text combined with auxiliary features and achieve a F_1-score of 44.6%. For subtask B, we employ a MLP neural network leveraging our estimates for subtask A and achieve a F_1-score of 30.1% (second place in the competition). We provide results and analysis of our system performance and present ablation experiments.

1 Introduction

Online social media has changed the way of communicating and disseminating media content and opinions, but also paved the way for spreading false or unverified rumors.

RumourEval 2019 (Gorrell et al., 2019) provides a dataset of labelled threads from Twitter and Reddit where each source post mentions a rumor. Subtask A (SDQC) consists of deciding for each post in a thread whether it is in a *support*, *deny*, *query*, or *comment* relation to the rumor. The goal of subtask B (Verification) it to classify the veracity of the rumor as *true*, *false*, or *unverified*. Figure 1 clarifies our terminology and the tasks.

Automated rumor classification is a challenging task as there is no definite evidence (e.g., authorized confirmation). In its absence, stance analysis is a useful approach. Systems that employ neural network architectures showed promising results in RumourEval 2017 (Derczynski et al., 2017), with

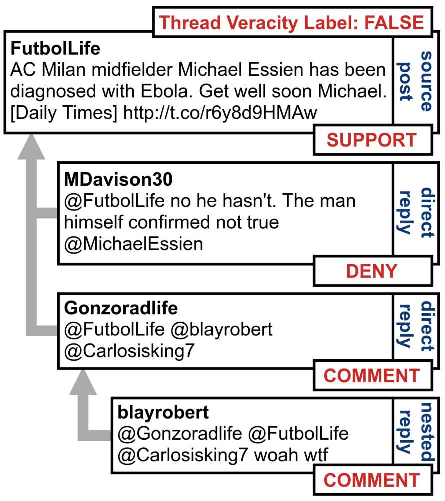

Thread Veracity Label: FALSE
FutbolLife
AC Milan midfielder Michael Essien has been diagnosed with Ebola. Get well soon Michael. [Daily Times] http://t.co/r6y8d9HMAw
source post
SUPPORT
MDavison30
@FutbolLife no he hasn't. The man himself confirmed not true @MichaelEssien
direct reply
DENY
Gonzoradlife
@FutbolLife @blayrobert @Carlosisking7
direct reply
COMMENT
blayrobert
@Gonzoradlife @FutbolLife @Carlosisking7 woah wtf
nested reply
COMMENT

Figure 1: An example Twitter thread from the training dataset with SDQC labels for each post and a veracity label for the thread's source post. Any post that does not reply to another is a *source post*. Reply posts can be *direct replies* (replies to a source post) or *nested replies* (replies that reply to another reply post). A *thread* is the set containing a source post and all its reply posts.

the LSTM-based sequential model of Kochkina et al. (2017) performing best.

In this paper, we describe our approach CLEARumor (ConvoLving ELMo Against Rumors) for solving both subtasks and provide empirical results and ablation experiments of our architecture. We make our PyTorch-based implementation and trained models publicly available[1].

2 System Description

After preprocessing the post text (Section 2.1) and embedding it with ELMo (Section 2.2), our architecture for subtask A (Section 2.3) passes the embedded text through a convolutional neural network (CNN) block, adds auxiliary features, and

[*]The first two authors contributed equally.

[1]https://github.com/Institute-Web-Science-and-Technologies/CLEARumor

uses a multilayer perceptron (MLP) block for estimating class membership. These estimates are combined with further auxiliary features and fed into an MLP block for the classification for subtask B (Section 2.4).

2.1 Preprocessing

For preprocessing, we rely mostly on Erika Varis Doggett's tokenizer for Twitter and Reddit[2], with which we strip away all user handles (e.g., "@FutbolLife"), remove the number sign in front of hash tags (e.g., "#Ebola" becomes "Ebola"), remove URLs, and limit repetitions of the same character to at most three times (e.g., "heeeeey" becomes "heeey"). We further decided to lowercase all text, which resulted in improved performance over mixed case in initial experiments. Last, all posts are truncated after 32 tokens[3].

2.2 ELMo Embeddings

The task of word embedding is to represent each word in a given sentence by a vector, which among other things allows for encoding words at the input layer in a neural network architecture. Traditional embedding methods such as word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014) work independently of context and always map the same word to the same vector.

In contrast, ELMo (Peters et al., 2018) is a recent embedding approach based on bidirectional LSTM networks that considers the context a word occurs in and is thereby able to address certain linguistic peculiarities, e.g., that the same word can have different meanings depending on its context. Further, ELMo incorporates subword units and is thereby able to represent words not seen during training successfully, an important benefit for the social media domain, where users frequently misspell existing words or introduce new ones. Formally, given a sequence of words $w_1 w_2 \ldots w_n$ ELMo represents the k-th word w_k as

$$\mathbf{ELMo}_k^{\text{task}} = \gamma^{\text{task}} \sum_{j=0}^{L} s_j^{\text{task}} \mathbf{h}_{k,j}, \qquad (1)$$

where L gives the number of internal layers that were used to train ELMo, $\mathbf{h}_{k,j}$ is the contextual

[2]https://github.com/erikavaris/tokenizer

[3]Only 10 out of the total 6634 Twitter posts are longer than this, while a few Reddit posts are up to 1,000 tokens long which would result in very impractical batch sizes.

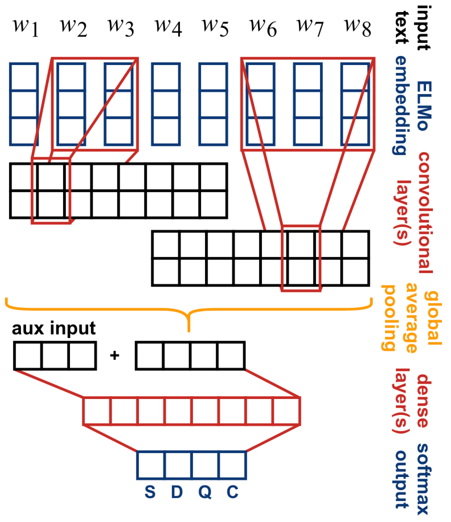


Figure 2: CLEARumor architecture for subtask A.

vector representation of layer j for word k, and γ^{task} and the s_j^{task} are scalars that can be tuned specifically for the task at hand.

We report results for the pretrained model `elmo_2x4096_512_2048cnn_2xhighway_5.5B`[4] for which $L = 2$ and which outputs 1024-dimensional embedding vectors (but didn't notice drastic improvements over the much smaller models). ELMo allows us to fine-tune γ^{task}, s_j^{task}, and even the $\mathbf{h}_{k,j}$ by backpropagating gradients to them, but we decided against this, because the RumourEval dataset is very small (cf. Table 1) adjusting these weights can quickly lead to overfitting, and keeping the weights constants allows us to precompute and store all ELMo embeddings once before the training process which results in a major boost in performance.

2.3 Subtask A

Our architecture for subtask A is visualized in Fig. 2. First, the tokenized text of the post that is to be classified as either support, deny, query, or comment is represented with an ELMo embedding. Next, the embedded text is fed into L_{conv}-many convolutional layers. Here, a single convolutional layers consists of multiple 1D-convolution operations with a set of different kernel sizes S, each mapping onto C convolutional channels, which are then concatenated along the channel axis. Each convolution operation is batch normalized (Ioffe and Szegedy, 2015) after a ReLU activation. To maintain an equal sequence length, sequences are padded with zero vectors. The result-

[4]https://allennlp.org/elmo

ing sequence representation is transformed into a single $|S| \cdot C$-dimensional vector via global average pooling. This sequence vector is concatenated with a vector of auxiliary features that encodes meta information about the post under classification (detailed in the next paragraph). Following is a stack of L_{dense}-many dense layers, for which dropout-regularization (Srivastava et al., 2014) is performed after ReLU activation. Finally, a single linear layer that is softmax-activated yields the four estimates of class membership. Parameters are optimized using Adam (Kingma and Ba, 2015) and a cross-entropy loss.

We use the following auxiliary features: (1) a two-dimensional Boolean vector encoding whether the post is from Twitter or Reddit; (2) a five-dimensional real-valued vector encoding meta-information for the post author: whether the user is verified or not, the number of followers they have, the number of accounts they follow themselves, and a ratio of the latter two numbers[5]; (3) the cosine similarity of the averaged ELMo embeddings of the post under classification to those of the thread's source post (defined to be 1 for source posts); and (4) a three-dimensional Boolean vector encoding whether the post is a source post, a direct reply, or a nested reply.

As hyperparameters, we employ a learning rate of 10^{-3}, a batch size of 512, and train for 100 epochs. In our loss function, we weigh the estimates of support, deny, and query equally but that of comment at only a fifth of the strength because of the imbalance of the dataset. We add L2-regularization with a weight of 10^{-2}. In our reported results we use $L_{\text{conv}} = 1$ convolutional layer, with kernel sizes $S = \{2, 3\}$ each mapping into $C = 64$ channels, after which follow $L_{\text{dense}} = 3$ dense layers with 128 hidden units each and a dropout of 0.5.

2.4 Subtask B

For subtask B we build a single feature vector that we feed into a MLP classifier. We reuse all the auxiliary features from subtask A except the last two, because all posts under classification in subtask B are source posts. We further add the following features: (1) a two-dimensional Boolean vector encoding whether media (an image or a URL) is attached to the post, (2) the upvote-to-downvote ratio of the post for Reddit (manually set to 0.5 for Twitter), (3) a two-dimensional real-valued vector encoding which fraction of the thread's posts are direct replies and which fraction are nested replies, (4) the averaged support, deny, and query probability estimates from subtask A averaged over all posts in the thread. Similarly to subtask A, this feature vector is fed into a stack of L_{dense}-many dense layers with dropout-regularization (Srivastava et al., 2014) after ReLU activation, after which a single softmax-activated linear layer yields estimates for the three classes true, false, or unverified.

Our model was trained with a learning rate of 10^{-3} and a batch size of 128 for 5000 epochs. In our loss calculation, we weigh the unverified class at 0.3 of the strength of the other two, and add a L2-regularization weight of 10^{-2}. We used $L_{\text{dense}} = 2$ dense layers with 512 hidden units each and a dropout of 0.25.

3 Evaluation

The dataset of RumourEval 2019 is summarized in Table 1. Our results for subtask A and B are

Subtask A		**S**	**D**	**Q**	**C**	**Σ**
Train	Twitter	910	344	358	2907	**5217**
	Reddit	15	34	37	612	
Dev	Twitter	94	71	106	778	**1485**
	Reddit	8	11	14	403	
Test	Twitter	141	92	62	771	**1827**
	Reddit	16	9	31	705	
Σ		**1184**	**561**	**608**	**6176**	**8529**

Subtask B		**T**	**F**	**U**	**Σ**
Train	Twitter	137	62	98	**327**
	Reddit	7	17	6	
Dev	Twitter	8	12	8	**38**
	Reddit	2	7	1	
Test	Twitter	22	30	4	**81**
	Reddit	9	10	6	
Σ		**185**	**138**	**133**	**456**

Table 1: Number of labelled instances for both subtasks of the RumourEval 2019 dataset broken down into (1) class frequencies, per (2) social media platform, and (3) training, development, and test dataset.

[5]We use min-max scaling based on the training data for these features. For Reddit the respective concepts don't exist and a vector of zeros is used instead.

Subtask A	**Dev**	**Test**					**CV**
	Macro-F$_1$	**Macro-F$_1$**	**S-F$_1$**	**D-F$_1$**	**Q-F$_1$**	**C-F$_1$**	**Macro-F$_1$**
Always Comment	22.1	22.3	0.0	0.0	0.0	89.4	—
Submitted	41.3	37.4	**46.7**	0.0	11.7	**91.2**	—
CLEAR$^{\text{aux}}$	**44.8**±0.6	42.7±0.6	29.6±0.6	**17.8**±2.4	43.9±1.0	79.5±1.3	47.1±4.5
CLEAR$^{\text{aux}}_{\text{MLP}}$	42.2±1.2	40.7±1.6	30.7±2.7	0.0±0.0	**51.6**±3.2	80.5±2.7	44.7±4.2
CLEAR$_{\text{CNN+MLP}}$	39.7±2.0	39.0±2.2	16.2±2.3	14.8±3.4	41.0±6.7	84.0±2.6	43.3±4.5
CLEAR$^{\text{aux}}_{\text{CNN+MLP}}$	42.9±2.2	**44.6**±2.6	34.6±3.7	15.4±3.1	42.2±8.3	86.1±1.1	**47.2**±3.8

Table 2: Evaluation results for subtask A. All reported scores are multiplied by 100. We provide the macro-averaged F$_1$-score for the development (Dev), the test (Test) datasets and for 10-fold cross validation (CV). For the test dataset, we further provide the individual F$_1$-scores per class. "Always Comment" is a baseline predicting always the most common class. "Submitted" are the results we officially submitted to RumourEval 2019. For our CLEARumor architecture we provide multiple ablation experiments. CLEAR$^{\text{aux}}_{\text{CNN+MLP}}$ is our full system, CLEAR$_{\text{CNN+MLP}}$ the same but without the auxiliary features, CLEAR$^{\text{aux}}_{\text{MLP}}$ instead uses no convolutional layers, and CLEAR$^{\text{aux}}$ just concatenates averages ELMo embeddings with auxiliary features uses a single linear layer.

Subtask B	**Dev**		**Test**		**CV**	
	Macro-F$_1$	**RMSE**	**Macro-F$_1$**	**RMSE**	**Macro-F$_1$**	**RMSE**
Submitted	41.7	0.743	28.6	0.764	—	—
CLEAR$_{\text{Subtask-B}}$	35.4±0.5	**0.676**±0.005	**30.1**±0.8	**0.754**±0.005	**26.7**±13.4	**0.733**±0.113
CLEAR$_{\text{NileTMRG}}$	**53.5**	0.761	18.6	0.846	—	—

Table 3: Evaluation results for subtask B. We report F$_1$ (multiplied by 100) and RMSE (root mean squared error) scores for the development (Dev), the test (Test) datasets and for 10-fold cross validation (CV). CLEAR$_{\text{Subtask-B}}$ is our subtask B architecture using the subtask A estimates from CLEAR$^{\text{aux}}_{\text{CNN+MLP}}$. CLEAR$_{\text{NileTMRG}}$ uses the same estimates but computes task B results using the NileTMRG system (Enayet and El-Beltagy, 2017).

detailed in Table 2 and Table 3, respectively.

The reported results differ from our official submission, because we continued to tune hyperparameters afterwards. We report results as trained on the training dataset and then evaluated on the development and test datasets, as provided by the RumourEval organizers. Because neural network experiments are naturally nondeterministic (Reimers and Gurevych, 2018) and we did indeed notice huge variances when retraining models, we report the mean and standard deviation over 10 runs for each experiment. Additionally, we report scores from a 10-fold cross validation over the whole dataset. Simple cross-validation would be inappropriate in our setting, because for example a split could result in the case where the same rumors occur in both the training and the test dataset which would allow a model to just memorize which posts are rumorous. We ensure that this does not happen in our case, by keeping all posts belonging to the same rumor[6] in the same cross validation fold. Note that scores on the organizer split and the cross validation are not directly comparable as different fractions of the whole dataset are used for training (~60-70% for the organizer split and ~90% for 10-fold cross validation).

4 Conclusion

We have presented CLEARumor, our architecture for the RumourEval 2019 shared tasks. In future we aim to generalize our approach, e.g., we currently use domain-specific features for characterizing the post author popularity, such as number of followers for Twitter, which are not available for all social media platforms. Besides investigating how well our approach translates to other languages, we are interested in studying the results for other pretrained word representation approaches, e.g., BERT (Devlin et al., 2018).

[6]For Twitter posts, the dataset contains rumor-topic labels for each thread, so we ensure that each topic only occurs in one fold. For Reddit posts, no labelling is available, so we can only ensure that all posts of a thread occur in the same fold.

Acknowledgments

Ipek Baris works for the EU-Project Co-Inform which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 770302. Lukas Schmelzeisen is supported by the German Research Foundation (DFG) as part of the Open Argument Mining project (grant STA 572/18-1).

References

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. SemEval-2017 Task 8: RumourEval: Determining rumour veracity and support for rumours. In *SemEval@ACL*, pages 69–76. ACL.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, abs/1810.04805.

Omar Enayet and Samhaa R. El-Beltagy. 2017. NileTMRG at SemEval-2017 Task 8: Determining Rumour and Veracity Support for Rumours on Twitter. In *SemEval@ACL*, pages 470–474. ACL.

Genevieve Gorrell, Kalina Bontcheva, Leon Derczynski, Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2019. SemEval-2019 Task 7: RumourEval: Determining Rumour Veracity and Support for Rumours. In *SemEval@NAACL-HLT*. ACL. To appear.

Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Elena Kochkina, Maria Liakata, and Isabelle Augenstein. 2017. Turing at SemEval-2017 Task 8: Sequential Approach to Rumour Stance Classification with Branch-LSTM. In *SemEval@ACL*, pages 475–480. ACL.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*, pages 1532–1543. ACL.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *NAACL-HLT*, pages 2227–2237. ACL.

Nils Reimers and Iryna Gurevych. 2018. Why Comparing Single Performance Scores Does Not Allow to Draw Conclusions About Machine Learning Approaches. *CoRR*, abs/1803.09578.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958.

Columbia at SemEval-2019 Task 7: Multi-task Learning for Stance Classification and Rumour Verification

Zhuoran Liu* **Shivali Goel*** **Mukund Yelahanka Raghuprasad**
Department of Computer Science, Columbia University
{zhuoran.liu, shivali.goel, my2541}@columbia.edu

Smaranda Muresan
Data Science Institute, Columbia University
sm761@columbia.edu

Abstract

The paper presents Columbia team's participation in the SemEval 2019 Shared Task 7: RumourEval 2019. Detecting rumour on social networks has been a focus of research in recent years. Previous work suffered from data sparsity, which potentially limited the application of more sophisticated neural architecture to this task. We mitigate this problem by proposing a multi-task learning approach together with language model fine-tuning. Our attention-based model allows different tasks to leverage different level of information. Our system ranked 6th overall with an F1-score of 36.25 on stance classification and F1 of 22.44 on rumour verification.

1 Introduction

The ubiquity of social media is allowing unverified news and rumours to spread easily. Efforts have been made to explore automated methods for rumour detection and verification (Derczynski et al., 2017; Zubiaga et al., 2016, 2018), and has shown promising potential to tackle this issue at scale.

RumourEval 2019 Shared Task 7 tackles the problem of predicting the veracity of rumours and stance of replies. It consists of two subtasks: task A (SDQC), in which stance (support, deny, querying, comment) of responses to a rumourous statement are predicted, and task B (verification), in which the statement's veracity is to be predicted. Size of training data provided for Task A is 5,217 and for Task B is 327.

In this paper, we proposed several methods to alleviate data sparsity and unleash the power of sophisticated neural models:

1. *Jointly learning to perform rumour verification and stance detection.* Training a neural network on limited amount of data for a single task is hard, especially in a sentence classification task. This is because of the weak supervision signal caused by the information asymmetry between the source text and the target labels. With supervision signal from multiple tasks, a neural network can exploit information in the training data more thoroughly.

2. *Using self-attention.* To predict the stance of a post, we want to selectively pay attention to some other posts that are relevant to this post. We use a QKV-style attention (Query, Key, Value) (Vaswani et al., 2017) to summarize the post context into a single vector (where in practice one attention head is usually enough). In addition, we use representations at different levels for different tasks.

3. *Using language model fine-tuning for stance classification.* We use the Universal Language Model Fine-tuning (ULMFiT) (Ruder and Howard, 2018) to improve our stance classifier. We begin with a generic language model trained on the Wikitext 103 dataset (Merity et al., 2016). This dataset consists of a large collection of pre-processed English Wikipedia articles. This enables the language model to properly model the general properties of language. Next, we fine-tune this language model on task specific data: RumourEval2019 dataset. Finally, a classification layer is added and the model is initialized with parameters from the fined-tuned language model.

Our system, which relies on these three key factors, are now publicly available.[2]

* Equal contribution.

[2]Github repository: https://github.com/joelau94/rumour2019-experiments

2 Related Work

Rumour Detection. Recently there has been a growing interest on developing methods for the task of rumour detection (Zubiaga et al., 2018), including a shared task in 2017 (Derczynski et al., 2017), which established a strong baseline for stance classification — task A(Kochkina et al., 2017), while (Enayet and El-Beltagy, 2017) established the same for veracity — task B. Dungs et al. (2018) discuss how stance information can facilitate veracity classification, while (Zubiaga et al., 2017) explore the use of contextual information for rumour detection. These results show that stance information and context information are important for rumour verification.

Multi-task Learning. Text classification tasks invariably suffer from the weak supervision signal due to loss of information in projecting text to task labels. There has been a growing number of works that explore multi-task learning for text classification (Zhang et al., 2017; Xiao et al., 2018). For the task of rumour detection specifically, there were attempts in jointly train for stance classification and rumour verification (Kochkina et al., 2018). Our muti-task approach uses a different, more advanced sentence embedding approach and uses the same LSTM for both tasks but with hidden states from different levels, which can be considered as different level representations of sentences. Empirically we found that higher levels of representation performs better for stance classification, while lower levels are better for veracity classification.

Transfer Learning with pre-trained Language Models. To alleviate the problem of data scarcity, researchers have proposed various approaches for pre-training language models on large-scale monolingual corpora, such as ELMo, ULMFiT, BERT, GPT, and have shown their effectiveness on several NLP tasks (Peters et al., 2018; Ruder and Howard, 2018; Devlin et al., 2018; Radford, 2018). In our work we use ULMFiT (Ruder and Howard, 2018) for stance classification.

3 System Description

We propose two system configurations:

1. *System1:* A joint-learning for task A and task B without using language model fine-tuning.
2. *System2:* Language model fine-tuning for task A.

3.1 System1: Joint Training for Stance Classification and Rumour Verification

We formulate the joint learning of Task A and B as follows: Given a branch of conversation $\mathbf{X}$ containing n posts

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n),$$

where each post $\mathbf{x}_k$ is a sequence of m_k words:

$$\mathbf{x}_k = (x_{k,1}, x_{k,2}, \cdots, x_{k,m_k}).$$

The goal is to build two neural probabilistic models $p(\mathbf{y}_{\text{stance}}|\mathbf{X}; \theta, \phi_{\text{stance}})$ and $p(y_{\text{veracity}}|\mathbf{X}; \theta, \phi_{\text{veracity}})$, where θ is the shared parameters, ϕs are parameters unique to each task, $\mathbf{y}_{\text{stance}} = (y_1, y_2, \cdots, y_n)$ are stance labels, and y_{veracity} is the veracity label.

To estimate θ and ϕs, we perform maximum likelihood estimation (MLE) over the training dataset $\mathcal{D} = \{(\mathbf{X}_d, \mathbf{y}_d)\}_{d=1}^{N}$, with optimization objectives being negative log-likelihoods:

$$\begin{aligned}
\mathcal{J}_{\text{stance}} &= -\sum_d \log p(\mathbf{y}_{\text{stance}}|\mathbf{X}; \theta, \phi_{\text{stance}}) \\
&= -\sum_d \sum_i \log p(y_i)_{\text{stance}}|\mathbf{X}; \theta, \phi_{\text{stance}}) \\
\mathcal{J}_{\text{veracity}} &= -\sum_d \log p(y_{\text{veracity}}|\mathbf{X}; \theta, \phi_{\text{veracity}})
\end{aligned}$$

Rumour verification and stance classification are highly-related tasks that can potentially provide useful information for each other. Therefore we integrate the two tasks for joint training, allowing more accurate estimation of the shared part of parameters θ.

To find an appropriate balance between the supervision signals from the two tasks, we introduce a tunable hyper-parameter λ. We then rewrite our objective function as follows:

$$\mathcal{J} = \lambda \cdot \mathcal{J}_{\text{stance}} + (1 - \lambda) \cdot \mathcal{J}_{\text{veracity}}$$

We designed an effective neural network to model $p(\mathbf{y}_{\text{stance}}|\mathbf{X}; \theta, \phi_{\text{stance}})$ and $p(y_{\text{veracity}}|\mathbf{X}; \theta, \phi_{\text{veracity}})$, which provides latent structures to capture subtleties of conversations. This model architecture is described below.

Neural Network Architecture

Inspired by the idea of BranchLSTM (Kochkina et al., 2017), we propose a model based on a single branch from the conversation tree. Our model is different from BranchLSTM (Kochkina et al., 2017) in the following ways:

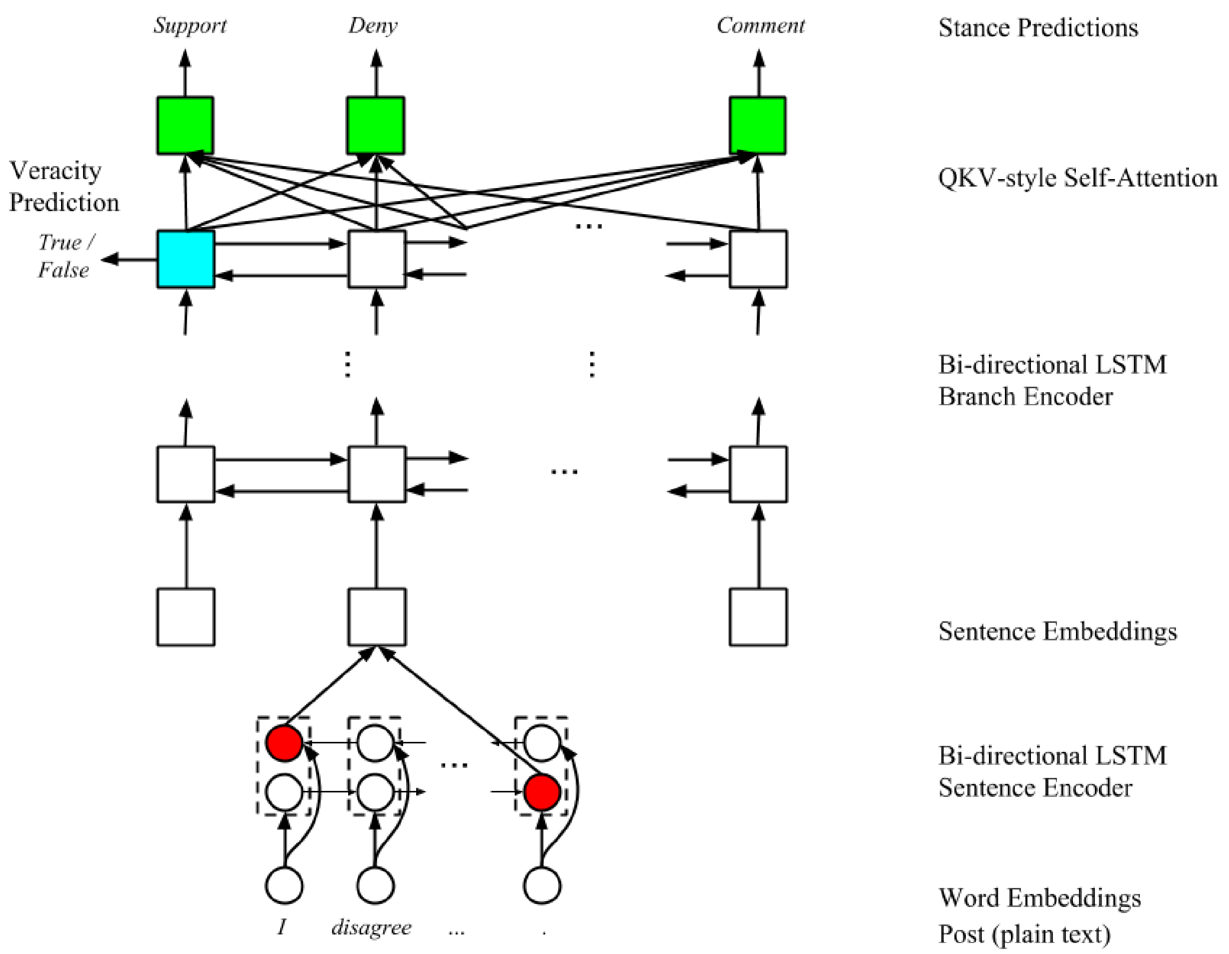


Figure 1: Model Architecture.

1. The sentence vector representation (sentence embedding) is generated with a bi-directional LSTM, as compared to simply taking the average of word vectors in BranchLSTM. This allows sentence embeddings to selectively encode important words and capture long-distance dependency in the sentence.

2. We apply a Transformer-style attention (Vaswani et al., 2017) on top of branch-level LSTM. This enables the most important information to flow in when trying to decide the stance of a post.

3. Rumour verification is incorporated as a task being jointly learnt together with stance classification, yet exploiting information at a different level from stance classification. In practice, hidden states at different levels of LSTM is being used for different tasks.

Figure 1 shows our overall model architecture, which we describe in more detail below.

Word Embeddings. The word embedding space is adjustable in our model. We initialize the word embedding matrix with pre-trained GloVe embeddings (Pennington et al., 2014). While we fix most word embedding vectors, we also keep some of the most frequent word embeddings trainable, allowing the word embedding space to adjust itself.

Sentence Embeddings. We consider each post as a sentence and we encode it with a bi-directional LSTM. We then take the last hidden state of the forward LSTM and first hidden state of the backward LSTM and concatenate them. The resulting vector can be considered as a dynamically generated sentence embedding.

Stacked Branch Encoder. To capture the interaction between posts in a branch of conversation, we use a stacked Bi-LSTM to encode the sentence embeddings obtained from previous steps. This results in a higher level representation of each post, which is fully aware of the conversation context. The higher the level in the LSTM stack, the more the representation is aware of context.

Attention. To predict the stance of a post, we want to selectively pay attention to some other posts that are relevant to this post. We use a QKV-style of attention (Vaswani et al., 2017) to summarize the post context into a single vector (where in practice one attention head is usually enough).

Stance Classifier. We first send the highest-level representation of posts to a QKV-style self-attention, which produces an attention-weighted context vector for each post in a branch. We then concatenate each post's representation with its corresponding context vector, and feed it through an MLP followed by a softmax for stance classification. During our experiment, we found that one attention head is good enough and is better than using multi-head attention.

Veracity Classifier. We take the representation of the original post (which is the first post in each conversation) from some intermediate layer, and feed it through an MLP followed by a softmax for veracity classification. This design corresponds to the intuition that when judging the authenticity of a post, the model should focus more on the post itself and less on how people judge it.

All hyperparameters can be found in our code with default settings.

3.2 System2: LM fine-tuning for Stance Classification

We also tried improving our stance classifier by using the Universal Language Model Fine-tuning (ULMFiT). After training a generic language model trained on Wikitext 103 dataset, we fine-tuned the LM on RumourEval2019 dataset.

Pre-processing was inspired by BranchLSTM system (Kochkina et al., 2017). Tweets along a particular branch were concatenated starting from the source tweet till the target node and considered as one training instance. The SDQC label of the last node concatenated was considered to be the label of the training instance. Here, replies are being referred to as nodes. For instance, source + reply 1 + reply 2, label of reply 2 was one training instance.

Finally, we used a BiLSTM max pooling network which was presented in (Conneau et al., 2017) and is shown in Figure 3. This model was initialized with parameters from the fined tuned language model. In this architecture, the representation generated by the BiLSTM was max pooled, i.e. the maximum value over each dimension was selected to form a fixed-size vector and was followed by softmax for stance classification.

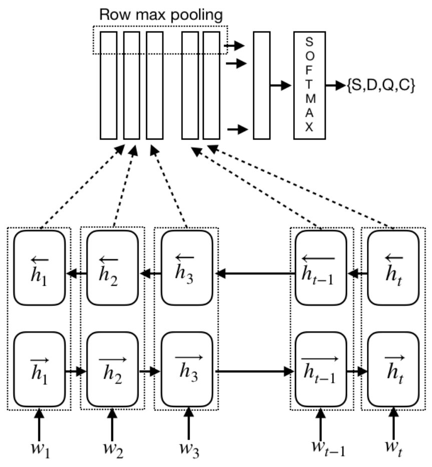


Figure 2: Max pooling in System2

4 Results and Analysis

For System1, we achieved an F1-score of 22.44 on task B and an F1-score of 34.04 on task A. For System2, we achieved F1-score of 36.25 on task A (System2 is only applied to task A). Performance of System1 on task A is slightly lower than the performance of System2, because we only treat task A as an auxiliary task for task B and did not apply ULMFiT to task A. Our final submission consisted of using System 2 for task A and System1 for task B. Our final submission ranked 6th in the final leaderboard.

	Verif	RMSE	SDQC
System 1	0.2244	0.8623	0.3404
System 2			0.3625
Final Submission	0.2244	0.8623	0.3625

Table 1: Performance of two systems on test set.

Unbalanced Class Labels. The model in System1 suffers heavily from an unbalanced class problem. From Table 2, we can see that the model is not giving any predictions of D (Deny) and Q (Query), which is why even though it has high accuracy (83%+), its F1 is lower than that of System2.

	S	D	Q	C
System 1	81	0	0	1746
System 2	62	16	84	1665

Table 2: Predicted class frequencies of SDQC classification on test data.

This problem is mitigated a little bit in System2, as we witnessed a few examples of D and Q predictions. This could potentially be because of the general knowledge gained by pre-training on large-scale Wikipedia text. Even then, D and Q classes are rare in the model predictions.

Instability in Training. System1 shows a perturbing training loss after it decrease to a certain level. After a certain point, the F1 score and accuracy on development set begins decreasing. One explanation is that the size of training data is too small and noise in the data negatively impacts the model.

5 Conclusion

In this work, we present the Columbia Team's system submission for the RumourEval 2019 shared task. We tackle the issue of data sparsity by multi-task learning which fully utilizes the training data. In addition, we also apply pre-training techniques such as ULMFiT which was effective in improving results on task A.

References

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. Semeval-2017 task 8: Rumoureval: Determining rumour veracity and support for rumours. In *SemEval@ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Sebastian Dungs, Ahmet Aker, Norbert Fuhr, and Kalina Bontcheva. 2018. Can rumour stance alone predict veracity? In *COLING*.

Omar Enayet and Samhaa R. El-Beltagy. 2017. Niletmrg at semeval-2017 task 8: Determining rumour and veracity support for rumours on twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 470–474, Vancouver, Canada. Association for Computational Linguistics.

Elena Kochkina, Maria Liakata, and Isabelle Augenstein. 2017. Turing at semeval-2017 task 8: Sequential approach to rumour stance classification with branch-lstm. In *SemEval@ACL*.

Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2018. All-in-one: Multi-task learning for rumour verification. In *COLING*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR*, abs/1609.07843.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Alec Radford. 2018. Improving language understanding by generative pre-training. In *Preprint*.

Sebastian Ruder and Jeremy Howard. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Liqiang Xiao, Honglun Zhang, and Wenqing Chen. 2018. Gated multi-task network for text classification. In *NAACL-HLT*.

Honglun Zhang, Liqiang Xiao, Yongkun Wang, and Yaohui Jin. 2017. A generalized recurrent neural architecture for text classification with multi-task learning. In *IJCAI*.

Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. 2018. Detection and resolution of rumours in social media: A survey. *ACM Comput. Surv.*, 51:32:1–32:36.

Arkaitz Zubiaga, Elena Kochkina, Maria Liakata, Rob Procter, and Michal Lukasik. 2016. Stance classification in rumours as a sequential task exploiting the tree structure of social media conversations. In *COLING*.

Arkaitz Zubiaga, Maria Liakata, and Rob Procter. 2017. Exploiting context for rumour detection in social media. In *SocInfo*.

GWU NLP at SemEval-2019 Task 7: Hybrid Pipeline for Rumour Veracity and Stance Classification on Social Media

Sardar Hamidian and Mona Diab
Department of Computer Science
The George Washington University
Washington DC, USA
{sardar, mtdiab} @gwu.edu

Abstract

Social media plays a crucial role as the main resource news for information seekers online. However, the unmoderated feature of social media platforms lead to the emergence and spread of untrustworthy contents which harm individuals or even societies. Most of the current automated approaches for automatically determining the veracity of a rumor are not generalizable for novel emerging topics. This paper describes our hybrid system comprising rules and a machine learning model which makes use of replied tweets to identify the veracity of the source tweet. The proposed system in this paper achieved 0.435 F-Macro in stance classification, and 0.262 F-macro and 0.801 RMSE in rumor verification tasks in Task7 of SemEval 2019.

1 Introduction

The number of users who rely on social media to seek daily news and information rises daily, but not all information online is trustworthy. The unmoderated feature of social media makes the emergence and diffusion of misinformation even more intense. Consequently, the propagation of misinformation online could harm an individual or even a society. Most of the current approaches on verifying credibility perform well for the unfold topics which are already verified by a trustworthy resource (Qazvinian et al., 2011; Hamidian and Diab, 2015). However, the performance suffers when it comes to real life application for dealing with the emerging rumors which are priorly unknown. Identifying the emerging rumor and veracity of the rumor by relying on previous observations is a challenging task as the new emerging rumor could be entirely new regarding the event, propagation pattern, and also the provenance. Despite these challenges, many researchers have been studying the generalizable metrics that could be aggregated from the source, replied posts, or network information (Vosoughi et al., 2018b; Kochkina et al., 2018; Grinberg et al., 2019). Our first mission in this paper is to automatically determine the veracity of rumors as part of the SemEval task. SemEval is an ongoing shared task for evaluations of computational sentiment analysis systems. Task 7 (RumourEval19) (Gorrell et al., 2019) is one of the twelve tasks, consisting of two subtasks. Task A is about stance orientation of people as supporting, denying, querying or commenting (SDQC) in a rumor discourse and Task B is about the verification of a given rumor. We propose a hybrid model with rules and a neural network machine learning scheme for both tasks. For task A we rely on the text content of the post, and its parent. In Task B not only do we aggregate contextual information of the source of the rumor but also using the veracity orientation of the others in the same conversation. We devise some rules to improve the performance of the model on query, deny, and support cases which are relatively essential classes in the verification tasks. Integrating the rule-based component we could reach a better performance in both tasks in comparison with a model which only relied on a machine learning approach.

2 Related work

There are several studies about the behavior of misinformation on social media, how it is distinguished and how social media users react to it. Most of these studies use data from Twitter since it has an infrastructure which allows researchers to access network information and meta information of all the users through Twitter APIs. In this section, we mainly focus on machine learning approaches in the study of rumor credibility and stance on Twitter.

One of the earliest work and the most relevant work to this task is that reported in Qazvinian et al. (2011), which addresses rumor detection (rumor/Not-rumor/undetermined) and opinion classification (deny/support/question/neutral) on Twitter using content-based as well as microblog-specific meme features. According to this work, content-based features performed better than meta information and network features for rumor identification and opinion classification tasks. In another study (Castillo et al., 2013), leveraged both information cascade and content features of the tweets by applying a supervised mechanism to identify credible and newsworthy content. According to Castillo's work "confirmed truth," or the rumors which are verified as true, are less likely to be questioned than false rumors regarding their validity. In mor recent study, (Vosoughi, 2015) proposes his two-step rumor detection and verification model on the Boston Marathon bombing tweets. The Hierarchical-clustering model is applied for rumor detection, and after the feature engineering process, which contains linguistic, user identity, and pragmatic features, the Hidden Markov model is applied to find the veracity of each rumor. Vosoughi (2015) also analyses the sentiment classification of tweets using the contextual Information, which shows how tweets in different spatial, temporal, and authorial contexts have, on average, different sentiments. In his recent work (Vosoughi et al., 2018a) he analyzed the spread of false and true news on Twitter on a large dataset. According to his research, fake news is more likely to diffuse deeper and longer in the information network than sound news. Moreover, his research suggests that false news are more novel and likely to be shared in comparison to the true news. Vosoughi et al. (2018a) also studied the false and true stories from an emotional perspective. According to his work, false stories inspired fear, disgust, and surprise in replies, while true stories inspired anticipation, sadness, joy, and trust.

3 Dataset

The dataset provided for this task contains Twitter and Reddit conversation threads associated with rumors about nine different topics on Twitter and thirty different topics on Reddit. The Ottawa shootings, Charlie Hebdo, the Ferguson unrest, Germanwings crash, and Putin missing are some of the rumors in this dataset. The overall size of the data including the development and evaluation set is 65 rumors on Reddit and 37 rumors with 381 conversations on Twitter. Table 1 illustrates all the information of underlying replies and source rumor in both social media platforms.

	Reddit		**Twitter**	
	#Src	*#Rep*	*#Src*	*#Rep*
Training	30	667	297	4222
Development	10	426	28	1021
Evaluation	25	736	56	1010
Total	65	1829	381	6253

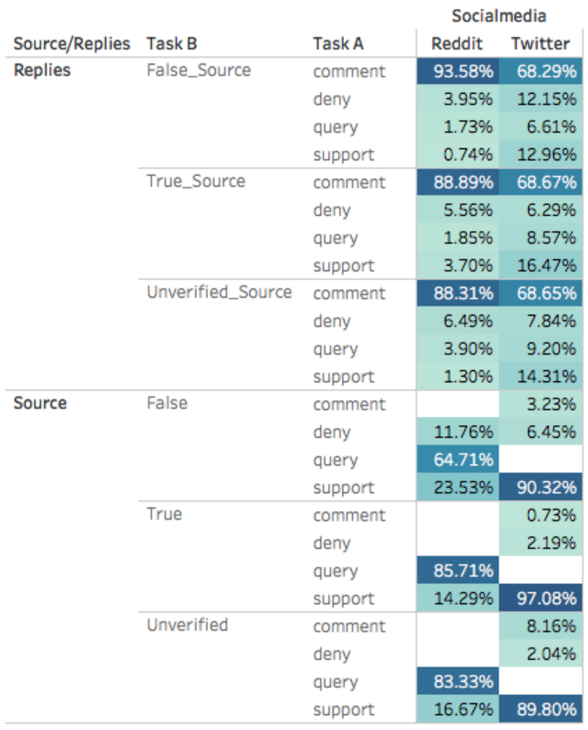
Table 1: Number of source (Src) conversations and replies (Rep) on Reddit and Twitter in the training, development and Evaluation sets.

3.1 Data insight

Figure 1 shows the distribution of the tags for both tasks across different platforms. According to the table, the stance orientation of the rumor conversations varies between Twitter and Reddit. In general, Reddit users leave more comments than Twitter users and this is regardless of the rumor veracity. In false rumors Twitter conversations are more oriented toward denial than Reddit's conversations; however, Twitter users support and deny false rumors to relatively the same extent. Twitter users are more supportive and ask more questions in regards to true rumors than the Reddit users, but they both deny true rumors to almost the same amount.
Interestingly, in both platforms, people question unverified rumors more than true and false rumors. For the source of conversation, Reddit and Twitter are significantly different. Regardless of the veracity, the source in Reddit conversations is more skewed to the query than the other stance tags, while Twitter is more toward the support. Despite some common characteristics Reddit and Twitter users behave differently when it comes to rumors. Reddit users do not deny the TRUE or UNVERIFIED rumors and question more when the rumor is false, yet Twitter users support more without any inquiries. It is worth noting that the conclusions mentioned in this section could only be valid for the data provided and in other conditions the same correlations might not be present.

4 System Description

For both tasks, we mainly rely on the content to determine the stance and verification of the



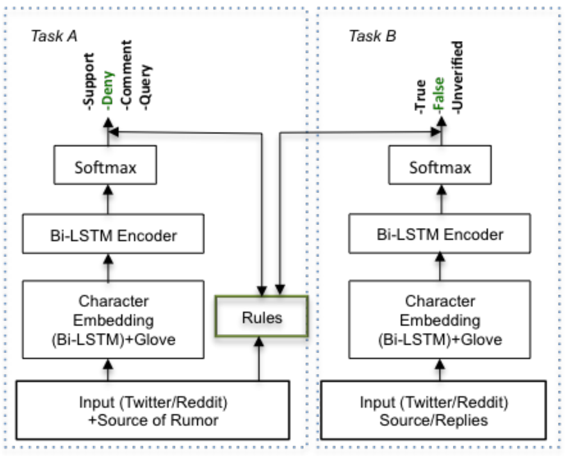
Source/Replies	Task B	Task A	Socialmedia Reddit	Twitter
Replies	False_Source	comment	93.58%	68.29%
		deny	3.95%	12.15%
		query	1.73%	6.61%
		support	0.74%	12.96%
	True_Source	comment	88.89%	68.67%
		deny	5.56%	6.29%
		query	1.85%	8.57%
		support	3.70%	16.47%
	Unverified_Source	comment	88.31%	68.65%
		deny	6.49%	7.84%
		query	3.90%	9.20%
		support	1.30%	14.31%
Source	False	comment		3.23%
		deny	11.76%	6.45%
		query	64.71%	
		support	23.53%	90.32%
	True	comment		0.73%
		deny		2.19%
		query	85.71%	
		support	14.29%	97.08%
	Unverified	comment		8.16%
		deny		2.04%
		query	83.33%	
		support	16.67%	89.80%

Figure 1: The distribution percentage of stance and verification tags on Twitter and Reddit dataset. "TaskB_Source" (exp. False_Source) indicate the verification tag of the source of conversation.

sources in the conversation. Our primary analysis in the insight section showed that there is a significant correlation between the two tasks. For the unverified rumors, the stance orientation is more toward queries rather than concluding support or denial; on the other hand, for true rumors, people are more likely to support or comment on the conversation than question or deny. Therefore, stance is key information to determine the veracity of the source rumor in the conversation. Task A is a four-way classification experiment in which we propose a hybrid model including a neural network-based (NN) model to encode the contextual representation of the post and its parent and then a rule-based model which is mainly designed to improve the performance on the minority classes including "support," "deny," and also "query." Task-B is a three-way classification task (True, False, and Unverified) in which we rely on both source and conversation content. We expand the veracity tags for the source of the conversation to the underlying posts and create a new set of veracity tags including Source_True, Source_False, and Source_Unverified (Six-way classification). We first apply a sequential neural network-based approach to identify the veracity tag of the source and also replied posts. From the sources with a low confidence value a voting mechanism is applied among all the posts in associated conversation, i.e. if the majority of the tweets in the conversation classified as Parent_true then the source of the conversation will be labeled as True.

4.1 Neural Network Approach

Given the success of recurrent neural networks (RNN) on language problems, we build a standard Bi-LSTM network for both tasks as illustrated in Figure 2. We also investigated the effectiveness of multitask learning in this experiment by sharing the information of two tasks in the same pipeline, but it does not lead to noticeable improvement in the performance.



Task A
-Support
-Deny
-Comment
-Query
Softmax
Bi-LSTM Encoder
Character Embedding (Bi-LSTM)+Glove
Rules
Input (Twitter/Reddit) +Source of Rumor
Task B
-True
-False
-Unverified
Softmax
Bi-LSTM Encoder
Character Embedding (Bi-LSTM)+Glove
Input (Twitter/Reddit) Source/Replies

Figure 2: Illustration of the hybrid network comprising the rule-based and Bi-LSTM-Softmax network on Task A and Task B.

4.1.1 Input Representations

Recent studies on NLP applications are reported to have good performance applying the pre-trained word embedding (Socher et al., 2013). We adopted two widely-used methods including the character embedding and pre-trained word vectors, i.e., GloVe (Pennington et al., 2014). We use a Bi-LSTM network to encode the morphology and word embeddings from characters. Intuitively the concatenated fixed size vectors $W_{Character}$ capture word morphology. $Word_{Character}$ is concatenated with a pre-trained word embedding from GloVe $W_{pre-trained-Glove}$ to get the final word representation. For Contextual Encoding, once the word embedding is created we use another Bi-LSTM layer to encode the contextual meaning from the sequence of word vectors $W_1, W_2, ..., W_t$

	Task A						Task B				
	Accuracy	Macro-F	Support	Query	Deny	Comment	Accuracy	Macro-F	True	False	Unverified
Dev	0.802	0.487	0.420	0.586	0.058	0.885	0.315	0.187	0.418	0.0	0.142
Test	0.796	**0.435**	0.446	0.408	0.0	0.886	0.382	**0.262**	0.525	0.0	0.260

Table 2: Accuracy and F score (macro-averaged) results on the development and test sets of Task A and B.

and consequently obtain a vector representation of a sentence from the final hidden state of the LSTM layer. The input representation would capture the word level syntax, semantics and contextual information. For Twitter data, we only rely on the tweet content for both source and replies, but for the Reddit rumor we use the "title" and "selftext" and only "body" for the replies.

4.2 Rule-based components

The first analysis of the data showed that stance knowledge could significantly help the determination of the source rumor in the conversation. However, due to imbalanced data, identifying the minority classes including deny, query, and support is challenging. We devise a new set of rules to improve the performance of Task A. Using the confidence values of the NN model we only selected the cases with low confidence for the rule-based experiments. We relied on simple rules for each stance class. For Query, a new set of rules was designed to identify the query cases using question marks and syntactic information of the sentence. For Deny, we calculated the cosine similarity of the source and response in addition to sentiment differences of the source and replied post. For the support cases, we mainly relied on the URL and picture existence in the content. The domain of the URL checked for being a fact-checking or news source. We also checked the existence of the picture in the post and consider that as one of the conditions for the supporting tweets.

5 Experimental Setup

The shared task dataset is split into training, development and test sets by the SemEval-2019 task organizers. We conducted and tuned the optimal set of hyperparameters by testing the performance on the development set and the output of the final model on the test set evaluated by the organizers. The statistics of the dataset are shown in Table 1.

5.1 Preprocessing

We applied various degrees of preprocessing on the content, we first removed the very short, deleted, and also the removed cases (Those that are labeled [deleted] or [removed] by the task organizers) from the dataset. We replaced the URLs from news sources with the token NURL and all the fact-checking URLs with FURLs. For compound words and hashtags, we used a simple heuristic. If the hashtag or a word contained an uppercase character in the middle of the word, then we split it before the uppercase letter. For instance, #PutinMissing are separating into two words Putin Missing.

5.2 Training

For all of the pipelines, the network is trained with backpropagation using Adam (Kingma and Ba, 2014), Root Mean Square Propagation (RmsProp), and Stochastic Gradient Descent (SGD) optimization algorithms. The parameters get updated in every training epoch. The character and Glove pre-trained embedding size [100, 200, 300] are examined with batch size 20 with 100 epochs. The training is stopped after no improvements in five consecutive epochs to ensure the convergence of the models. The highest performance on the development set was achieved under the following parameters: hidden size of Bi-LSTM (100); optimization (RMSprop); initial learning rate (0.003); L2 ($Lambda = 0.1$); character and word embedding size (300, 100); dropout size (0.3).

6 Result and Evaluation

In this section, we discuss the experimental results in both tasks. Table 2 shows overall and per category results for Task A and B. The proposed model achieved 0.435 F-Macro in stance classification, and 0.262 F-macro and 0.801 RMSE in rumor verification tasks. In overall evaluation, we ranked as the third group in Task B and tenth in Task A out of twenty-five teams.

7 Conclusion

Identifying rumor veracity is an important and challenging task. Our first mission in this paper is to automatically determine the veracity of rumors as part of the SemEval task. We proposed a hybrid model comprising the rules and NN machine learning approach to identify the stance in the rumor conversation and the veracity of the source in Twitter and Reddit datasets. The proposed system achieved the third best performance for RumourEval, Task7 of Semeval 2019.

—

References

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2013. Predicting information credibility in time-sensitive social media. *Internet Research*, 23(5):560–588.

Genevieve Gorrell, Kalina Bontcheva, Leon Derczynski, Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2019. SemEval-2019 Task 7: RumourEval: Determining rumour veracity and support for rumours. In *Proceedings of SemEval*. ACL.

Nir Grinberg, Kenneth Joseph, Lisa Friedland, Briony Swire-Thompson, and David Lazer. 2019. Fake news on twitter during the 2016 us presidential election. *Science*, 363(6425):374–378.

Sardar Hamidian and Mona Diab. 2015. Rumor Detection and Classification for Twitter Data. *SOTICS 2015: The Fifth International Conference on Social Media Technologies, Communication, and Informatics*, (c):71–77.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2018. All-in-one: Multi-task Learning for Rumour Verification.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Vahed Qazvinian, Emily Rosengren, Dragomir R Radev, and Qiaozhu Mei. 2011. Qazvinian et al. - 2011 - Rumor has it Identifying Misinformation in Microblogs(2). *Conference on Empirical Methods in Natural Language Processing*, pages 1589–1599.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Soroush Vosoughi. 2015. *Automatic detection and verification of rumors on Twitter*. Ph.D. thesis, Massachusetts Institute of Technology.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018a. The spread of true and false news online. *Science*, 359(6380):1146–1151.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018b. The spread of true and false news online. *Science*, 359(6380):1146–1151.

—

SINAI-DL at SemEval-2019 Task 7: Data Augmentation and Temporal Expressions

Miguel Ángel García-Cumbreras[1], **Salud María Jiménez-Zafra**[1], **Arturo Montejo-Ráez**[1], **Manuel Carlos Díaz-Galiano**[1], **Estela Saquete**[2]
[1]CEATIC / Universidad de Jaén
[1]{magc, sjzafra, amontejo, mcdiaz}@ujaen.es
[2]DLSI / Universidad de Alicante
[2]stela@dlsi.ua.es

Abstract

This paper describes the participation of the SINAI-DL team at RumourEval (Task 7 in SemEval 2019, subtask A: SDQC). SDQC addresses the challenge of rumour stance classification as an indirect way of identifying potential rumours. Given a tweet with several replies, our system classifies each reply into either supporting, denying, questioning or commenting on the underlying rumours. We have applied data augmentation, temporal expressions labeling and transfer learning with a four-layer neural classifier. We achieve an accuracy of 0.715 with the official run over reply tweets.

1 Introduction

Fake news has been identified as "news stories that have no factual basis but are presented as news" (Allcott and Gentzkow, 2017). On a conceptual level, we can define hoaxes or rumours as false information spread across social media with the intention to be picked up by traditional news websites (Rubin et al., 2015). Rumours have been around for millennia, as attested by the ancient (and modern) Greek word 'pheme', which means rumour or inaccurate information.

We have participated in RumourEval (SemEval 2019 Task 7, Subtask A), named *SDQC: Determining support for rumours*, with the complementary objective of tracking how other sources orient to the accuracy of the rumourous story by looking at the replies to the tweet that presented the rumourous statement (Gorrell et al., 2019).

These replies are extracted from Twitter and Reddit, but we have only processed the tweet replies, obtaining a good score in terms of accuracy.

The rest of the paper is organized as follows. Section 2 is a brief overview of the task. Data analysis is shown in section 3. Section 4 describes the neural network architecture. The experiments and results are analyzed in section 5. Finally, conclusions and proposals for further experimentation are provided in section 6.

2 Related Work

The previous edition of RumorEval was organized as part of the SemEval 2017 workshop. Thirteen systems were presented in that edition. Most of the systems presented face this task as a tweet classification task with four categories. Some participants use neural networks such as LSTM (Kochkina et al., 2017) and CNN (Chen et al., 2017; García Lozano et al., 2017), or SVM machine learning algorithm (Wang et al., 2017; Singh et al., 2017), using as main feature word embeddings. Most systems add lexical, syntactic and semantic features to word embeddings.

3 Data analysis

The data provided by the organization consist of a set of tweets and replies. Replies can be originated from two different sources: Reddit or Twitter. We have only worked with Twitter replies because features of Reddit replies and tweets are different, especially in regards to the length. In Table 1 we present the datasets distribution.

Dataset	Tweets	Replies	Tweets replies
train_EN	327	5,217	4,244
dev_EN	38	1,485	1,025
test_EN	56	1,746	1,010

Table 1: Datasets distribution (only tweets).

The objective of task A is to determine whether each reply supports, denies, queries or comments the rumour. The classification of tweet replies in each of the four categories established is shown in Table 2. We can conclude that although the labels show a realistic situation in terms of user comments, the classes are very unbalanced.

Category	train_EN	dev_EN	test_EN
Comment	2,897	779	771
Deny	335	70	92
Query	358	107	56
Support	634	69	91

Table 2: Tweets replies distribution.

In order to decide which window size to use in our system, we generated a cumulative histogram according to the different lengths of the tweets replies. Our objective was to select a size that could cover a high rate of tweets replies. In Table 3 we summarize the quantiles at 80 % and 90 % for the different datasets. Based on the values we decided to select a window size with 30 words because approximately 90 % of training and development tweet replies have a length of 30 words or less.

Data	Quantile 0.8	Quantile 0.9
train_EN	25	29
dev_EN	28	30

Table 3: Length of tweet replies covering 80 % and 90 % of cases.

4 System overview

Nowadays, deep neural architectures are populating the scientific playground in many scenarios: image recognition, speech recognition (Graves et al., 2013) and synthesis (Ze et al., 2013), and, of course, text classification (Zhang et al., 2015). But these supervised learning algorithms demand certain requirements that sometimes are difficult to meet. One of the most difficult to overcome in some cases is the need for a large and varied learning data set. When there is a lack of data, two main strategies can be followed: *transfer learning* and *data augmentation*.

4.1 Model description

We have implemented the proposed neural network using the Keras[1] library for Python, running on TensorFlow over an NVIDIA Titan X card. Each model took approximately 15 minutes to get trained and few seconds to classify development or test sets. The architecture of our neural network follows a sequence of layers as follows:

[1]http://keras.io

1. **First layer**: An embedding layer that is loaded with pre-trained weights, and converts each word into a 200-dimensional vector of real values.
2. **Second layer**: A bi-directional LSTM recurrent network with 512 activations and a dropout value of 0.5.
3. **Third layer**: A dense network with 128 activations and the *ReLU* function as activation function. A dropout of 0.5 is also applied after this network.
4. **Fourth layer**: last classification layer, with 4 activations on the final softmax function.

Figure 1 shows the neural network model with the four layers.

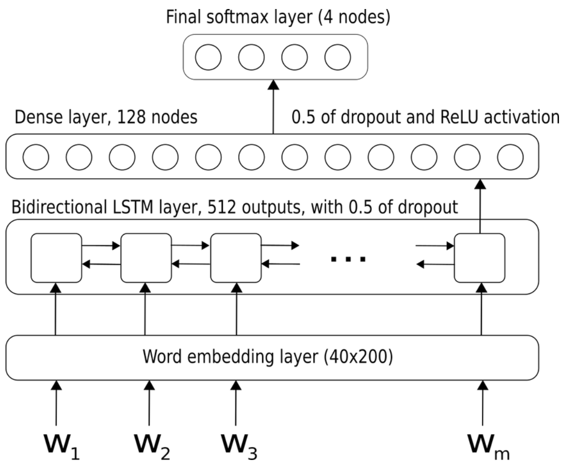

Final softmax layer (4 nodes)
Dense layer, 128 nodes
0.5 of dropout and ReLU activation
Bidirectional LSTM layer, 512 outputs, with 0.5 of dropout
...
Word embedding layer (40x200)
w_1
w_2
w_3
w_m

Figure 1: Neural network model.

The model has been trained with the hyperparameters values specified in Table 4 (ce means cross-entropy).

Parameter	value
Batch size	512
Loss function	categorical ce
Optimization algorithm	Adam
Sequence length	30 terms
No. Epochs	50

Table 4: Hyperparameters

The texts have been preprocessed as follows:

1. Lower case is applied.
2. Hashtags are split into several tokens according to a camel case approach. For example, "*#MeToo*" is converted into the terms "*<BOH>me too <EOH>*".

3. Mentions are replaced by the token <*MENTION*>

4. Unknown terms (those not found in the embedding dictionary) are replaced by the token <*UNK*>.

5. A final token <*EOT*> is added at the end of the tweet.

We have taken an already trained word embedding matrix for the first layer, allowing the weights of the these learned model to get retrained during the learning process. We have used the weights from the GloVe Twitter model provided by the Stanford NLP Group, which is built over 2 billion tweets (27B tokens, 1.2M vocab, uncased, 200-dimensional vectors, 1.42 GB download).

4.2 Data augmentation

There are two important reasons for proposing data augmentation. On the one hand, deep neural models require a significant amount of training data to extract relevant features. On the other hand, as we can see in the section 3 there is a strong class imbalance between the samples (in dev and train datasets more than 75 % of the tweets are labeled as *comments*).

For each tweet, our system expands the information using paraphrasing. To express the same message with different words, we applied the online tool RewriterTools[2]. For instance, the paraphrase of the tweet *"EU's hailed migrant plan 'a road to Hell' Czech Republic refuses involvement"* is *"EU's hailed migrant layout 'a avenue to Hell' Czech Republic refuses involvement"*.

4.3 Temporal expressions

As human beings, we tend to organize the flux in structured units known as events. Events take place at certain times, which are expressed in the text in the form of temporal expressions. However, these expressions are not always explicit dates that a computer is able to understand. For this reason, we decided to add a module that is capable of processing temporal information at the level of events and temporal expressions and annotate and resolve this information, so that it can be used in the detection of a rumor.

TimeML (Saurí et al., 2006) is the most standardized schema to annotate temporal information. They defined the event as "something that can be said to obtain or hold true, to happen or to occur". This annotation schema annotates not only events and temporal expressions, but also temporal relations, known as links (Pustejovsky et al., 2003). Example below shows a sentence annotated with TimeML temporal expressions (`TIMEX3`), events (`EVENT`), and the links between them (`TLINK`).

```
John <EVENT eid='e1'>came</EVENT> on <TIMEX3 tid='t1'>Monday</TIMEX3>
<TLINK eventInstanceID='e1'
relatedToTime='t1'
relType='IS_INCLUDED' />
```

In our approach, the Temporal Information Extraction and Processing was performed by TIPSem system (Temporal Information Processing using Semantics) (Llorens et al., 2013, 2012)[3]. TIPSem is able to automatically annotate all the temporal information according to TimeML standard annotation scheme (Saurí et al., 2006). In this first approach of the system, only the tags regarding temporal expressions and events have been considered and we will explore using the links as further work.

5 Experiments and results

We performed an evaluation of the proposed neural network on the development set, but training a model on two different official training sets: the official ones and those augmented by paraphrasing the given tweets. The results were discouraging when paraphrased tweets are added to the training set, as Table 5 shows. After checking the tweets generated by the paraphrasing tool, we noticed that the quality was low, with non-sense texts in some cases and few structural variations from the original tweet. Thus, we believe that the network was not even less robust, but worse as a non-realistic model was learned.

The detection of temporal expressions and the inclusion of the generated tags into the model didn't report any improvement either. We believe that the related embeddings (randomly initialized) needed a far larger set to fit in the transferred learned embedding model for GloVe vectors.

Thus, our submission relies only on official training data which was, as we know, not enough data to ensure a good learning process. Anyhow, our system performed in 9th position out from 21 in

[2]https://www.rewritertools.com/paraphrasing-tool

[3]`http://gplsi.dlsi.ua.es/demos/TIMEE/`

train data	accuracy on dev data
official	0.690499
official + paraphrased	0.675808
official with temporal tags	0.684622

Table 5: Development experiments

subtask A, with an SDQC value of 0.3927 (F1-score).

Table 6 shows the results obtained with the official run over test set (only with the tweet replies).

truth label	accuracy	total	correct
all labels	**0.7148**	**1,066**	**762**
support	0.0219	91	2
deny	0.1413	92	13
query	0.4285	56	24
comment	0.9377	771	723

Table 6: Test experiments: official run

In the first analysis of results we can verify that the neural network system, on the base case, has worked correctly (almost perfect) for the *comment* class that have a sufficient number of examples of train and dev, and much worse for those with very few examples (classes support, deny and query).

We have to finish a more exhaustive analysis of these results, especially of the mislabelled samples. For instance, in the analysis of the truth label *support*, our system tags the most of the cases with the *comment* label. In this case, we can conclude that the *comment* label has been overtrained because of the greater number of examples (high bias).

6 Conclusions and future work

Our proposal explores how transferred embeddings and data augmentation may help in a text classification task like RumourEval. By augmenting official training data with paraphrasing, no improvement is noticed on classifying development data, due to the poor quality of the paraphrasing tool. So, we plan to explore other augmentation strategies, like a forward-backward translation. Neither temporal expression detection has been found useful in this task, at least with the model proposed. We have found also that the models trained exhibits high variance. That means that we are overfitting the model on training data, so despite the use of the dropout technique, early stopping, fewer parameters or more training data could help to produce a more robust model. Attention mechanism in the neural network could also help (Wang et al., 2016), along with a pre-training of the LSTM with a large corpus of tweets for a language model (predicting next word) and then transfer those weights and retrain them for this specific task.

Finally, we intend to incorporate a module that takes into account the reputation of the user who makes comments, based on non-textual parameters, such as the relationship between the user of the original tweet and the user of the reply tweet, number of followers, knowledge of the subject, etc. We will use that information to work in the second task, predicting the veracity of the original tweet.

Acknowledgements

This research work is partially supported by a grant from the Ministerio de Educación Cultura y Deporte (MECD - scholarship FPU014/00983), the project REDES (TIN2015-65136-C2-1-R) and a grant from the Fondo Europeo de Desarrollo Regional (FEDER).

References

Hunt Allcott and Matthew Gentzkow. 2017. Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, 31(2):211–236.

Yi-Chin Chen, Zhao-Yang Liu, and Hung-Yu Kao. 2017. IKM at SemEval-2017 Task 8: Convolutional Neural Networks for stance detection and rumor verification. In *Proceedings of SemEval-2017*, pages 465–469, Vancouver, Canada. ACL.

Marianela García Lozano, Hanna Lilja, Edward Tjörnhammar, and Maja Karasalo. 2017. Mama Edha at SemEval-2017 Task 8: Stance Classification with CNN and Rules. In *Proceedings SemEval-2017*, pages 481–485, Vancouver, Canada. ACL.

Genevieve Gorrell, Kalina Bontcheva, Leon Derczynski, Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2019. SemEval-2019 Task 7: RumourEval: Determining rumour veracity and support for rumours. In *Proceedings of SemEval*. ACL.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

Elena Kochkina, Maria Liakata, and Isabelle Augenstein. 2017. Turing at SemEval-2017 Task 8: Sequential Approach to Rumour Stance Classification with

Branch-LSTM. In *Proceedings of SemEval-2017*, pages 475—480, Vancouver, Canada. ACL.

Hector Llorens, Estela Saquete, and Borja Navarro-Colorado. 2012. Automatic System for Identifying and Categorizing Temporal Relations in Natural Language. *International Journal of Intelligent Systems*, 27(7):680–703.

Hector Llorens, Estela Saquete, and Borja Navarro-Colorado. 2013. Applying Semantic Knowledge to the Automatic Processing of Temporal Expressions and Events in Natural Language. *Information Processing & Management*, 49(1):179–197.

James Pustejovsky, José M. Castaño, Robert Ingria, Roser Saurí, Robert J. Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R. Radev. 2003. Timeml: Robust specification of event and temporal expressions in text. In *New Directions in Question Answering, Papers from 2003 AAAI Spring Symposium, Stanford University, Stanford, CA, USA*, pages 28–34.

Victoria L. Rubin, Yimin Chen, and Niall J. Conroy. 2015. Deception detection for news: Three types of fakes. In *Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community*, ASIST '15, pages 83:1–83:4, Silver Springs, MD, USA. American Society for Information Science.

Roser Saurí, Jessica Littman, Robert Knippen, Robert Gaizauskas, Andrea Setzer, and James Pustejovsky. 2006. *TimeML Annotation Guidelines 1.2.1 (http://www.timeml.org/)*.

Vikram Singh, Sunny Narayan, Md Shad Akhtar, Asif Ekbal, and Pushpak Bhattacharyya. 2017. Iitp at semeval-2017 task 8 : A supervised approach for rumour evaluation. In *Proceedings of SemEval-2017*, pages 497–501, Vancouver, Canada. ACL.

Feixiang Wang, Man Lan, and Yuanbin Wu. 2017. ECNU at SemEval-2017 Task 8: Rumour Evaluation Using Effective Features and Supervised Ensemble Models. In *Proceedings of SemEval-2017*, pages 491–496, Vancouver, Canada. ACL.

Yequan Wang, Minlie Huang, Li Zhao, et al. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615.

Heiga Ze, Andrew Senior, and Mike Schuster. 2013. Statistical parametric speech synthesis using deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7962–7966. IEEE.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

UPV-28-UNITO at SemEval-2019 Task 7: Exploiting Post's Nesting and Syntax Information for Rumor Stance Classification

Bilal Ghanem[1], **Alessandra Teresa Cignarella**[1,2],
Cristina Bosco[2], **Paolo Rosso**[1], **Francisco Rangel**[1,3]
1. PRHLT Research Center, Universitat Politècnica de València
2. Dipartimento di Informatica, Università degli Studi di Torino
3. Autoritas Consulting
bigha@doctor.upv.es, cigna@di.unito.it,
prosso@dsic.upv.es, bosco@di.unito.it, francisco.rangel@autoritas.es

Abstract

In the present paper we describe the UPV-28-UNITO system's submission to the RumorEval 2019 shared task. The approach we applied for addressing both the subtasks of the contest exploits both classical machine learning algorithms and word embeddings, and it is based on diverse groups of features: stylistic, lexical, emotional, sentiment, meta-structural and Twitter-based. A novel set of features that take advantage of the syntactic information in texts is moreover introduced in the paper.

1 Introduction

The problem of rumor detection lately is attracting considerable attention, also considering the very fast diffusion of information that features social media platforms. In particular rumors are facilitated by large users' communities, where also expert journalists are unable to keep up with the huge volume of online generated information and to decide whether a news is a hoax (Procter et al., 2013; Webb et al., 2016; Zubiaga et al., 2018).

Rumour stance classification is the task that intends to classify the type of contribution to the rumours expressed by different posts of a same thread (Qazvinian et al., 2011) according to a set of given categories: supporting, denying, querying or simply commenting on the rumour. For instance, referring to Twitter, once a tweet that introduces a rumour is detected (the "source tweet"), all the tweets having a reply relationship with it, (i.e. being part of the same thread), are collected to be classified.

Our participation to this task is mainly focused on the investigation of linguistic features of social media language that can be used as cues for detecting rumors[1].

2 Related work

The RumorEval 2019 shared task involves two tasks: Task A (rumour stance classification) and Task B (verification).

Stance Detection (SD) consists in automatically determining whether the author of a text is in favour, against, or neutral towards a given target, i.e. statement, event, person or organization, and it is generally indicated as TARGET-SPECIFIC STANCE CLASSIFICATION (Mohammad et al., 2016).

Another type of stance classification, more general-purpose, is the OPEN STANCE CLASSIFICATION task, usually indicated with the acronym SDQC, by referring to the four categories exploited for indicating the attitude of a message with respect to the rumour: Support (S), Deny (D), Query (Q) and Comment (C) (Aker et al., 2017). Target-specific stance classification is especially suitable for analyses about a specific product or political actor, being the target given as already extracted, e.g. from conversational cues. On this regard several shared tasks have been organized in recent years: see for instance SemEval-2016 Task 6 (Mohammad et al., 2017) considering six commonly known targets in the United States, and StanceCat at IberEval-2017 on stance and gender detection in tweets on the matter of the Independence of Catalonia (Taulé et al., 2017). On the other hand, the open stance classification, (i.e. the task addressd in this paper), is more suitable in

[1]Source code is available on GitHub: https://github.com/bilalghanem/UPV-28-UNITO

classifying emerging news or novel contexts, such as working with online media or streaming news analysis.

Provided that attitudes around a claim can act as proxies for its veracity, and not only of its controversiality, it is reasonable to consider the application of SDQC techniques for accomplishing rumour analysis tasks. A first shared task, concerning SDQC applied to rumor detection, has been organized at SemEval-2017, i.e RumorEval 2017 (Derczynski et al., 2017). Furthermore, several research works have analyzed the open issue of the impact of rumors in social media (Resnick et al., 2014; Zubiaga et al., 2015, 2018), for instance exploiting linguistic features (Ghanem et al., 2018). Such a kind of approaches may be also found in works which deal with the problems of Fake News Detection (Ciampaglia et al., 2015; Hanselowski et al., 2018).

Furthermore, a rumor is defined as a "circulating story of questionable veracity, which is apparently credible but hard to verify, and produces sufficient scepticism and/or anxiety so as to motivate finding out the actual truth" (Zubiaga et al., 2015).

Concerning veracity identification, increasingly advanced systems and annotation schemas have been developed to support the analysis of rumour veracity and misinformation in text (Qazvinian et al., 2011; Kumar and Geethakumari, 2014; Zhang et al., 2015).

3 Description of the task

The RumorEval task is articulated in the following sub-tasks: **Task A** (open stance classification – SDQC) is a multi-class classification for determining whether a message is a "support", a "deny", a "query" or a "comment" wrt the original post; **Task B** (verification) is a binary classification for predicting the veracity of a given rumour into "true" or "false" and according to a confidence value in the range of 0-1.

3.1 Training and Test Data

The RumourEval 2019 corpus contains a total of 8,529 English posts, namely 6,702 from Twitter and 1,827 from Reddit.

The portion of data from Twitter has been built by combining the RumorEval 2017 training and development datasets (Derczynski et al., 2017), and includes **5,568** tweets: 325 source tweets (grouped into eight overall topics such as Charlie Hebdo attack, Ottawa shooting, Germanwings crash...), and 5,243 discussion tweets collected in their threads.
The dataset from Reddit, which has been instead newly released this year, is composed by **1,134** posts: 40 source posts and 1,094 collected in their threads.

	Training	Test
Twitter	5,568	1,066
Reddit	1,134	761
Total	6,702	1,827

Table 1: Training and test data distribution.

All data have been split in training and test set with a proportion of approximately $80\% - 20\%$ (see Table 1).

4 UPV-28-UNITO Submission

The approach and the features selection we applied is the same for both tasks and is based on a set of manual features described in Section 4.1. We built moreover another set of features (i.e. second-level features) extracted by using the manual features together with features based on word embeddings (see Section 4.2 for a detailed description). For modeling the features distribution with respect to each thread, we used for task B the same features as in task A. Then, in both tasks, we fed the features to a classical machine learning classifier.

4.1 Manual Features

For enhancing the selection of features, we investigated the impact of diverse groups of them: emotional, sentiment, lexical, stylistic, meta-structural and Twitter-based. Furthermore, we introduced a novel set of syntax-based features.

Emotional Features - We exploited several emotional resources in order to build features for our system. Three lexica: (a) **EmoSenticNet**, a lexicon that assigns six WordNet Affect emotion labels to SenticNet concepts (Poria et al., 2013); (b) the **NRC Emotion Lexicon**, a list of English words and their associations with eight basic emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive) (Mohammad and Turney, 2010);

and (c) **SentiSense**, an easily scalable concept-based affective lexicon for Sentiment Analysis (De Albornoz et al., 2012). We also exploited two tools: (d) **Empath**, a tool that can generate and validate new lexical categories on demand from a small set of seed terms (Fast et al., 2016); and (e) **LIWC** a text analysis dictionary that counts words in psychologically meaningful categories (Pennebaker et al., 2001).

Sentiment Features - Our sentiment features were modeled exploiting sentiment resources such as: (a) **SentiStrength**, a sentiment strength detection program which uses a lexical approach that exploits a list of sentiment-related terms (Thelwall et al., 2010); (b) **AFINN**, a list of English words rated for valence with an integer between minus five (negative) and plus five (positive) (Nielsen, 2011); (c) **SentiWordNet**, a lexical resource in which each WordNet synset is associated to three numerical scores, describing how objective, positive, and negative the terms contained in the synset are (Esuli and Sebastiani, 2007); (d) **EffectWordNet**, a lexicon about how opinions are expressed towards events, which have positive or negative effects on entities (+/-effect events) (Choi and Wiebe, 2014); (e) **SenticNet**, a publicly available resource for opinion mining built exploiting Semantic Web techniques (Cambria et al., 2014); and (f) the **Hu&Liu** opinion lexicon[2].

Lexical Features - Various lexical features already explored in similar Sentiment Analysis tasks were employed: (a) the presence of **Bad Sexual Words**, a list extracted from the work of Frenda et al. (2018); (b) the presence of **Cue Words** related to the following categories: *belief, denial, doubt, fake, knowledge, negation, question, report* (Bahuleyan and Vechtomova, 2017); the categories *an, asm, asf, qas, cds* of the multilingual hate lexicon with words to hurt **HurtLex** (Bassignana et al., 2018); (d) the presence of **Linguistic Words** related to the categories of *assertives, bias, fatives, implicatives, hedges, linguistic words, report verbs*; (e) the presence of specific categories present in **LIWC**: *sexual, certain, cause, swear, negate, ipron, they, she, he, you, we, I.* (Pennebaker et al., 2001).

Stylistic Features - We employed canonical stylistic features, already thoroughly explored in Sentiment Analysis tasks and already proven useful in multiple domains: (a) the count of **question marks**; (b) the count of **exclamation marks**; (c) **length** of a sentence; (d) the **uppercase ratio**; (e) the count of consecutive **characters** and **letters**[3] (f) and the presence of **URLs**.

In addition to the above-listed, common features exploited in Sentiment Analysis tasks, in this work we introduce two novel sets of features: (1) **Problem-specific features** (considering the fact that the dataset is composed by Twitter data and Reddit data) and (2) **Syntactical features**.

Meta-structural features - Since training and test data are from Twitter and Reddit both, we explored meta-structural features suitable for data coming from both platforms: (a) the **count of favourites/likes**, in which we have two different value distribution (Twitter vs. Reddit), so we normalized them in a range 0-100; (b) the **creation time** of a post, encoded in seconds; (c) the **count of replies**; and (d) the **level**, i.e. the degree of "nestedness" of the post in the thread.

Twitter-only Features - Because of the duplicitous nature of the RumorEval 2019 dataset (Twitter and Reddit), some of the several features, already thoroughly used in Sentiment Analysis tasks and based on Twitter metadata, could not be used in this task[4]. As follows: (a) the presence of **hashtags**; (b) the presence of **mentions**; (c) the count of **retweets**. And also some user-based features: (d) whether the user is **verified** or not; (f) the count of **followers**; (g) the count of **listed** (i.e. the number of public lists of which this user is a member of); (h) the count of **statuses**; (i) the count of **friends** (i.e. the number of users that one account is following); (l) the count of **favourites**.

Syntactic Features - In our system some feature has been also modeled by referring to syntactic information involved in texts (Saif et al., 2016). After having parsed[5] the dataset in the *Universal De-*

[2]http://www.cs.uic.edu/liub/FBS

[3]We considered 2 or more consecutive characters, and 3 or more consecutive letters.

[4]For the instances from Reddit, that did not have a representation of one of the following features, the empty values has been filled with a weighted average of the values obtained by other similar instances.

[5]The parsing system we applied is UDPipe, available at: https://pypi.org/project/ufal.udpipe/

pendency[6] format, thus obtaining a set of syntactic "dependency relations" (*deprel*), we were able to exploit: (a) the **ratio of negation** dependencies compared to all the other relations; (b) the Bag of Relations (BoR_all) considering all the *deprels* attached to **all the tokens**; (c) the Bag of Relations (BoR_list) considering all the *deprels* attached to the tokens belonging to a selected **list of words** (from the lists already made explicit in the paragraph "Lexical Features" in Section 4.1); and finally (d) Bag of Relations (BoR_verbs) considering all the *deprels* attached to all the **verbs**, thus fully exploiting morpho-synctactic knowledge.

4.2 Second-level Features

For the second-level features, we employed (a) the cosine similarity of one instance wrt its parents and (b) information of the tree structure of a thread, exploiting its "nesting" and depth from the source tweet.

Similarity with Parents - In this feature, we used the cosine similarity to measure the similarity between each post with its parents. The parents of a reply are the (A) direct upper-level post and (B) the source post in the thread (see Figure 1).

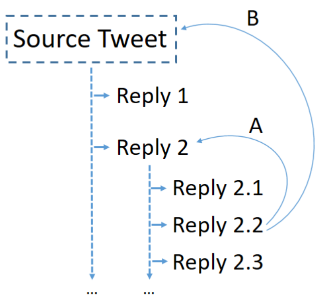


Figure 1: An example for reply 2.2 parents.

We extracted the cosine similarity in A and B by using the manual features' final vector and words embeddings average vectors of the posts; the words embeddings average vector for a post is extracted by averaging the embeddings of the post's words[7].

[6]The *de facto* standard for the representation of syntactical knowledge in the NLP community: https://universaldependencies.org/

[7]We used the pre-trained Google News word embeddings in our system: https://code.google.com/archive/p/word2vec/

SDQC Depth-based Clusters - We built level-based stance clusters from the posts. For each stance class (SDQC), we extracted all the belonging posts that correspond to one of the four classes and we computed the average value of the feature vectors (as one unique cluster). Since we have four main stances, this process ended with four main clusters. For the feature extraction, we measured the cosine similarity for each post wrt these four clusters. As done in the previous feature described above, we built these clusters by using both the manual features' vectors and word embeddings' vectors of the posts, so each stance cluster is represented in two ways. In these four main clusters, we didn't consider the nesting of the posts in the thread.

Also, we obtained the same clusters but instead of averaging all the posts that correspond to a stance, we considered the nesting of the posts in the thread. We split the nesting of the threads into five groups: posts with depth one, two, three, four, five or larger. For each of these levels, we extracted four SDQC clusters (depth-based). For instance, if a post occurs in depth two, we measured the cosine similarity between this post and 1) the four main SDQC clusters[8], 2) the four depth-based SDQC clusters two.

Concerning task B, we modeled the distribution of the features used for task A. For each thread we did the following:

1. We counted how many posts in the thread correspond to each of the stances.
2. We extracted the averaged features' vectors for each stance's posts in the thread.
3. We extracted the standard deviation for each stance's posts in the thread.

5 Experiments

We tested different machine learning classifiers in each task performing 10-fold cross-validation. The results showed that the Logistic Regression (LR) produces the highest scores. For tuning the classifier, we used the Grid Search method. The parameters of the LR are: $C = 61.5$, $penalty = L2$, and since the dataset is not balanced, we used different weights for the classes as COMMENT =

[8]Four features using the manual features, and another four using the words embeddings.

0.10, DENY = 0.35, SUPPORT = 0.20 and QUERY = 0.35. We conducted an ablation test on the features employed in task A in order to investigate their importance in the classification process. Table 2 presents the ablation test results as well as the system performance using 10-fold cross-validation.

SET	FEATURE	M-F1
A	All features	54.9
B	A - Emotional features	54.5
C	A - Sentiment features	54.7
D	A - Lexical features	53.6
E	A - Syntactic features	54.7
F	A - Stylistic features	50.1
G	A - Meta-structural features	54.5
H	A - Twitter-only features	54.9
I	A - Cosine similarity with parents	55.3
I.1	I using only manual features	54.9
I.2	I using only words embeddings	54.9
J	A - SDQC depth-based clusters	47.7
J.1	J using only manual features	53.3
J.2	J using only words embeddings	51.1
K	A-(C+E+I)	55.6
L	A-(B+C+E+G)	55.7
M	A-(B+C+E+G+I.2)	55.9

Table 2: Ablation test.

Provided that the organizers allowed two submissions for the final evaluation, on both tasks we used all the features (set A) in the first submission and set M for the second submission. In Table 3 we present the final scores achieved on both tasks.

	MACRO-F1	RMSE
Task A	48.95	–
Task B	19.96	82.64

Table 3: Final results.

6 Error Analysis

A manual error analysis allow us to see which categories and posts turned out to be the most difficult to be dealt with our system. We found out that SUPPORT was misclassified 114 times, DENY 92 times, QUERY 44 times, and COMMENT 57 times. Therefore, SUPPORT seems to be the hardest category to be correctly classified.
Table 4 reports the detailed confusion matrix of predicted vs. gold labels and shows that the most of errors are related to the category SUPPORT (in the gold dataset) and COMMENT (in our runs), while any error involves the more contrasting classes (e.g. SUPPORT and DENY). By better investigating the gold test set, it should be moreover observed that several semantically empty messages of the test set have been marked using some class, while our system marks them as COMMENT, i.e. selecting the more frequent class when a clear indication of the content is lacking.

		PREDICTED			
		S	D	Q	C
GOLD	S	–	0	13	101
	D	1	–	6	85
	Q	5	1	–	38
	C	5	17	35	–

Table 4: Confusion matrix of errors.

7 Conclusion

In this paper we presented an overview of the UPV-28-UNITO participation for *SemEval 2019 Task 7 - Determining Rumour Veracity and Support for Rumours*.

We submitted two different runs in the detection of rumor stance classification (Task A) and veracity classification (Task B) in English messages retrieved from Twitter and Reddit both. Our approach was based on emotional, sentiment, lexical, stylistic, meta-structural and Twitter-based features. Furthermore, we introduced two novel sets of features, i.e. *syntactical* and *depth-based* features, which proved to be successful for the task of rumor stance classification, where our system ranked as 5th (out of 26) and, according to the RMSE score, we ranked 6th in Task B for veracity classification. Since the two latter groups of features produced an interesting contribution to the score for Task A, but they were fairly neutral in Task B, we will follow this trail and try to inquire more on these aspects in our future work.

Acknowledgments

The work of Cristina Bosco was partially funded by Progetto di Ateneo/CSP 2016 (*Immigrants, Hate and Prejudice in Social Media*, S1618L2BOSC01).

References

Ahmet Aker, Leon Derczynski, and Kalina Bontcheva. 2017. Simple open stance classification for rumour analysis. *arXiv preprint arXiv:1708.05286*.

Hareesh Bahuleyan and Olga Vechtomova. 2017. UWaterloo at SemEval-2017 Task 8: Detecting

Stance Towards Rumours with Topic Independent Features. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 461–464.

Elisa Bassignana, Valerio Basile, and Viviana Patti. 2018. Hurtlex: A Multilingual Lexicon of Words to Hurt. In *5th Italian Conference on Computational Linguistics, CLiC-it 2018*, volume 2253, pages 1–6. CEUR-WS.

Erik Cambria, Daniel Olsher, and Dheeraj Rajagopal. 2014. SenticNet 3: a Common and Common-sense Knowledge Base for Cognition-driven Sentiment Analysis. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*.

Yoonjung Choi and Janyce Wiebe. 2014. +/-effectwordnet: Sense-level Lexicon Acquisition for Opinion Inference. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1181–1191.

Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. 2015. Computational Fact Checking from Knowledge Networks. *PloS one*, 10(6).

Jorge Carrillo De Albornoz, Laura Plaza, and Pablo Gervás. 2012. SentiSense: An Easily Scalable Concept-based Affective Lexicon for Sentiment Analysis. In *LREC*, pages 3562–3567.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. SemEval-2017 Task 8: RumourEval: Determining Rumour Veracity and Support for Rumours. *arXiv preprint arXiv:1704.05972*.

Andrea Esuli and Fabrizio Sebastiani. 2007. SentiWordNet: a High-coverage Lexical Resource for Opinion Mining. *Evaluation*, 17:1–26.

Ethan Fast, Binbin Chen, and Michael S. Bernstein. 2016. Empath: Understanding Topic Signals in Large-scale Text. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 4647–4657. ACM.

Simona Frenda, Bilal Ghanem, and Manuel Montes-y Gómez. 2018. Exploration of Misogyny in Spanish and English Tweets. In *3rd Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018)*, volume 2150, pages 260–267. CEUR-WS.

Bilal Ghanem, Paolo Rosso, and Francisco Rangel. 2018. Stance Detection in Fake News A Combined Feature Representation. In *Proceedings of the 1st Workshop on Fact Extraction and VERification (FEVER)*, pages 66–71.

Andreas Hanselowski, Avinesh PVS, Benjamin Schiller, Felix Caspelherr, Debanjan Chaudhuri, Christian M Meyer, and Iryna Gurevych. 2018. A Retrospective Analysis of the Fake News Challenge Stance Detection Task. *arXiv preprint arXiv:1806.05180*.

KP Krishna Kumar and G Geethakumari. 2014. Detecting misinformation in online social networks using cognitive psychology. *Human-centric Computing and Information Sciences*, 4(1):14.

Saif M. Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. SemEval-2016 Task 6: Detecting Stance in Tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41.

Saif M. Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. 2017. Stance and Sentiment in Tweets. *ACM Transactions on Internet Technology*, 17(3):26:1–26:23.

Saif M. Mohammad and Peter D. Turney. 2010. Emotions Evoked by Common Words and Phrases: Using Mechanical Turk to Create an Emotion Lexicon. In *Proceedings of the NAACL HLT 2010*, pages 26–34. ACL.

Finn Årup Nielsen. 2011. A new ANEW: Evaluation of a Word List for Sentiment Analysis in Microblogs. *arXiv preprint arXiv:1103.2903*.

James W. Pennebaker, Martha E. Francis, and Roger J. Booth. 2001. Linguistic Inquiry and Word Count: LIWC 2001. *Mahway: Lawrence Erlbaum Associates*, 71.

Soujanya Poria, Alexander Gelbukh, Amir Hussain, Newton Howard, Dipankar Das, and Sivaji Bandyopadhyay. 2013. Enhanced SenticNet with Affective Labels for Concept-based Opinion Mining. *IEEE Intelligent Systems*, 28(2):31–38.

Rob Procter, Farida Vis, and Alex Voss. 2013. Reading the Riots on Twitter: Methodological Innovation for the Analysis of Big Data. *International Journal of Social Research Methodology*, 16(3):197–214.

Vahed Qazvinian, Emily Rosengren, Dragomir R Radev, and Qiaozhu Mei. 2011. Rumor has it: Identifying Misinformation in Microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1589–1599. ACL.

Paul Resnick, Samuel Carton, Souneil Park, Yuncheng Shen, and Nicole Zeffer. 2014. Rumorlens: A System for Analyzing the Impact of Rumors and Corrections in Social Media. In *Proceedings of the Computational Journalism Conference*.

Hassan Saif, Yulan He, Miriam Fernandez, and Harith Alani. 2016. Contextual Semantics for Sentiment Analysis of Twitter. *Information Processing & Management*, 52(1):5–19.

Mariona Taulé, Maria Antònia Martí, Francisco M. Rangel Pardo, Paolo Rosso, Cristina Bosco, and Viviana Patti. 2017. Overview of the Task on Stance and Gender Detection in Tweets on Catalan Independence. In *Proceedings of the 2nd Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2017)*, volume 1881, pages 157–177. CEUR-WS.org.

Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment Strength Detection in Short Informal Text. *Journal of the American Society for Information Science and Technology*, 61(12):2544–2558.

Helena Webb, Pete Burnap, Rob Procter, Omer Rana, Bernd Carsten Stahl, et al. 2016. Digital Wildfires: Propagation, Verification, Regulation, and Responsible Innovation. *ACM Transactions on Information Systems (TOIS)*, 34(3):15.

Qiao Zhang, Shuiyuan Zhang, Jian Dong, Jinhua Xiong, and Xueqi Cheng. 2015. Automatic detection of rumor on social network. In *Natural Language Processing and Chinese Computing*, pages 113–122. Springer.

Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. 2018. Detection and Resolution of Rumours in Social Media: A Survey. *ACM Computing Surveys (CSUR)*, 51(2):32.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Kalina Bontcheva, and Peter Tolmie. 2015. Towards Detecting Rumours in Social Media. In *AAAI Workshop: AI for Cities*.

BLCU_NLP at SemEval-2019 Task 8: A Contextual Knowledge-enhanced GPT Model for Fact Checking

Wanying Xie, Mengxi Que, Ruoyao Yang, Chunhua Liu, Dong Yu(✉)
Beijing Language and Culture University, Beijing, China
{xiewanying07, quemengxi, yangruoyao97, chunhualiu596}@gmail.com
yudong@blcu.edu.cn

Abstract

Since the resources of Community Question Answering are abundant and information sharing becomes universal, it will be increasingly difficult to find factual information for questioners in massive messages. SemEval 2019 task 8 is focusing on these issues. We participate in the task and use Generative Pretrained Transformer (OpenAI GPT) as our system. Our innovations are data extension, feature extraction, and input transformation. For contextual knowledge enhancement, we extend the training set of subtask A, use several features to improve the results of our system and adapt the input formats to be more suitable for this task. We demonstrate the effectiveness of our approaches, which achieves 81.95% of subtask A and 61.08% of subtask B in accuracy on the SemEval 2019 task 8.

1 Introduction

With the development of Community Question Answering (cQA) forums, massive information is being shared. However, not all information is factual, which makes finding an appropriate answer to satisfy the information needs of questioners more difficult. Previous work which concentrated on these problems (Nakov et al., 2017) reranked the questions based on their relevance with the original question. Šaina et al. (2017) treated the similarity ranking task as a binary classification problem. We study these issues in SemEval-2019 Task 8 (Mihaylova et al., 2019) by using the contextual Knowledge-enhanced GPT (Radford et al., 2018), which use Transformer (Vaswani et al., 2017) as model architecture. The contextual knowledge enhancement includes data extension, feature extraction, and input transformation.

The task includes two subtasks and they are both three classification problems. In subtask A, we need to find out whether a question seeks a factual answer, an opinion or just want to socialize with others. We classify the answers for questions that look for factual information in subtask A into three classes in subtask B: true, false or nonfactual. In this paper, we study both subtasks and use a similar system to solve them.

Several challenges exist when doing this task. The size of datasets for both subtasks is small. The data contains a number of complex long text, which makes extracting key information more difficult. The input format of GPT changes with different objectives of tasks, so it requires some modifications to fit specific tasks.

We apply three points to solve these problems. We extend the training set of subtask A from two other datasets: DailyDialog (Li et al., 2017) and SQuAD2.0 (Rajpurkar et al., 2018). We use two methods to guarantee the quality of expanded datasets. Firstly we use the Levenshtein Distance to screen similar data, and then we use the prediction of the model to further screen the results of the previous step. Goyal (2017) and Xie et al. (2017) used various features. Le et al. (2017) used keywords to solve the previous similar problem. We follow their work in feature extraction. Working on subtask A, we also use characteristic words as features to improve the system. Input transformation for classification task is $Start + Text + Extract$, including randomly initialized start and end tokens. We concatenate the text and features token sequences with a delimiter token.

The remainder of this paper is organized as follows. Section 2 contains a description of our system. The experiments and analysis of the results are introduced in section 3. We describe the conclusions in section 4.

2 System Description

As Figure 1 shows, our system is composed of the following components: data extension, feature extraction, input transformation, and model.

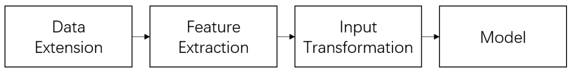


Figure 1: Flowchart of the system

Since the data provided by the task organizers are insufficient, our model does not get a high accuracy on such a small amount of data. We apply the data extension(Section 2.1) to address this problem. The extended datasets are DailyDialog and SQuAD2.0. We use two Levenshtein Distance and model prediction to ensure the expanded data similar enough to original data.

Since the data is composed of long text with complex information, it is difficult for our model to extract key information. We use feature extraction(Section 2.2) to solve this problem, which is able to bring high discrimination between data categories. We add two kinds of features to the input: original features directly extracted from the training data and observed features is summarized by us. After feature extraction, the key information gets enhanced and our model has easier access to significant information of data.

Different types of tasks correspond to different input transformation of GPT. We change the input transformation(Section 2.3) to fit different tasks. If a task is about classification, the input format is supposed to be $Start + Text + Extract$. We need to adjust the input of our model to adapt to the specific task, also need to add features in the data. We use special character $Delim$ to connect different features and the main text, then we use the connected formats as our input.

2.1 Data Extension

This section briefly introduces the datasets of subtask A, subtask B, DailyDialog and SQuAD2.0. Then we introduce data extension from DailyDialog and SQuAD2.0. Figure 2 shows the total process of extending data for subtask A. We use two approaches to ensure the data that we expanded from other datasets similar enough with original data. The two approaches are Levenshtein Distance screening and model prediction screening.

Data Overview The datasets we used for subtask A and subtask B are provided by the task organizers. We extend the training set of subtask A from two other datasets: DailyDialog and SQuAD2.0.

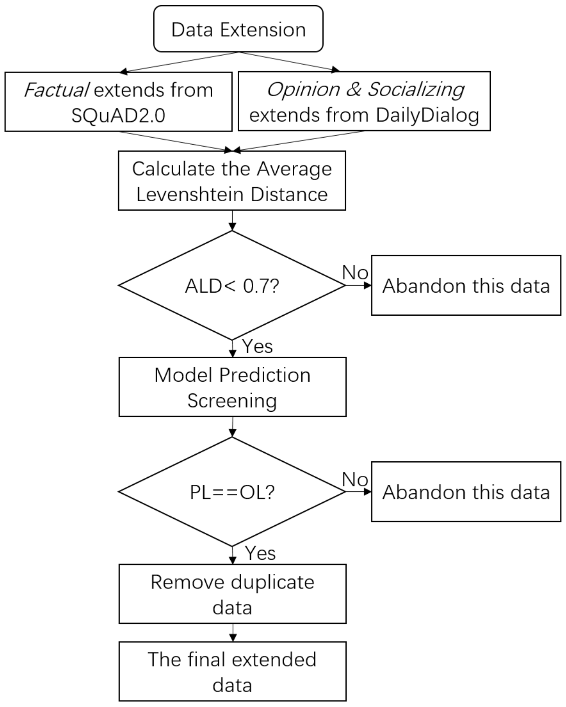

Data Extension
Factual extends from SQuAD2.0
Opinion & Socializing extends from DailyDialog
Calculate the Average Levenshtein Distance
ALD< 0.7?
No
Abandon this data
Yes
Model Prediction Screening
PL==OL?
No
Abandon this data
Yes
Remove duplicate data
The final extended data

Figure 2: Flowchart of data extension. ALD represents average Levenshtein distance. PL stands for perdicted label and OL stands for original label. Since *Opinion* and *Socializing* are both extended in Daily Dialog, so we remove duplication in the alternative extended datasets of *Opinion*.

- In subtask A, there are 1118 threads in the original dataset. Each thread consists of RelQBody, RelQSubjcet, RelQCategory, RelQDate, RelQId, RelQUser Id and RelQUsername. RelQBody is a complete question description but its text is too long. RelQSubject is short but lacks feature information. RelQCategory is mentioned to show the question's category and it is useful in classifying the questions. Our emphasis is subtask A, which is working on deciding the questions' classification label *Factual*, *Opinion* and *Socializing*. Specifically, the size of each class is 311, 563 and 244. The *Opinion* class accounts for more than half of the total data.

- The dataset of Subtask B is similar to subtask A. There are 495 answers in the original dataset. One question corresponds to one or several answers. The purpose of subtask B is to divide the answers into three categories: *True*, *False* and *Nonfactual*.

- DailyDialog is a multi-turn dialog dataset, which includes questions and answers, and the data size is 11318. The topic of DailyDialog is about daily life, so we consider that we expand *Opinion* and *Socializing* label data from it.

- SQuAD2.0 is a reading comprehension dataset, which intends to answer a question according to the context. Absolutely, questions in this dataset all ask for factual information since the answers can be found from the context. We decided to extend the *Factual* label data from the questions of this dataset.

Levenshtein Distance Screening Levenshtein distance is also known as Editing distance, which refers to the smallest number of editing operations required to change one string into another. The editing operation consists of three choices: replacing one character with another, inserting one character, and deleting one character. When comparing the two sentences, they will be more similar if the Levenshtein distance is smaller.

Levenshtein distance is used to calculate the similarity between two sentences. Since the length of sentences in the dataset is uncertain, the Levenshtein distance is an integer of indeterminate size. We divide the Levenshtein distance by a larger length of two sentences. Ultimately, what we get is not an unlimited integer, but a decimal between 0 and 1. In this paper, it is called average Levenshtein distance, as shown in Equation(1), where ALD(s1,s2) represents the average Levenshtein distance of sentence 1 and sentence 2.

$$ALD(s1, s2) = \frac{LevenshteinDistance}{max(len(s1), len(s2))} \quad (1)$$

We regard the question data of the cQA(community QA) forum as the original data and divide it into three categories according to the different labels of the questions. When traversing instance in *Opinion* and *Socializing*, the ALD between the original data and the question of DailyDialog data is calculated. When traversing the data of *Factual*, the ALD between the original data and the question of SQuAD 2.0 data is calculated.

We set a threshold of **0.7**. At this threshold, we are able to get more data which is guaranteed to be sufficiently similar. After calculating the ALD, if the value is less than the threshold, it means the two sentences are sufficiently similar. Then we use this data as alternative extended data. Finally, we get three alternative extended datasets with their original label. Since *Opinion* and *Socializing* are both extended from DailyDialog, it will be some plication in the two extended datasets. The original *Socializing* data is less than *Opinion*, so we remove duplication in the extended datasets of *Opinion* in order to get roughly the same amount of data. It means if a data appears in the alternative extended datasets of *Socializing*, then remove this data in the alternative extended datasets of *Opinion*.

Model Prediction Screening The question dataset of the cQA forum is used as the training set to train the GPT model, and the candidate extended dataset is used as the test set to predict. If the predicted label is consistent with the original label of the test set, then this data is considered to be correct prediction data. As one of the extended datasets, if the predicted label is inconsistent with the original label, which means that it is wrong, it is considered that this data is not helpful to the model, so we discard this data.

	original	extended	sum
factual	311	434	745
opinion	563	308	871
socializing	244	598	842

Table 1: Class distribution of subtask A

After screening by Levenshtein distance and model prediction, we finally get 434, 308 and 598 for *Factual*, *Opinion* and *Socializing* to expand. After expansion, the number of three classified data is 745, 871 and 842, respectively. Table 1 shows the total data category distribution of the subtask A.

2.2 Feature Extraction

We introduce two kinds of features and explain how we apply these features to the GPT model in detail.

Features Acquisition Subtask A includes two kinds of features. One we call *Original Features* is given directly in the dataset. The other we call *Observed Features* is obtained from the data observation. For subtask B, we just use the features

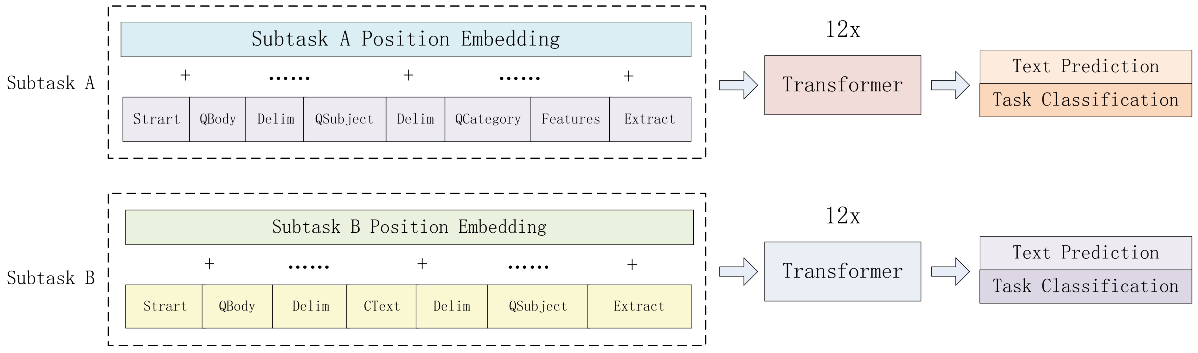


Figure 3: Input formats of subtasks. The input formats consist of text embedding and position embedding. Subtasks A and B input to the model separately. In the model, it is a 12-layer decoder-only transformer with masked self-attention heads. Text prediction and task classification are both the fine-tuning objective of the GPT model.(Radford et al., 2018)

extracted directly in the dataset. We present two kinds of features as follows.

Original Features In the dataset of subtask A, each thread consists of RelQBody, RelQSubjcet, RelQCategory, RelQDate, RelQId, RelQUser Id and RelQUsername. The main text is RelQbody, and we consider other information as features. Through our screening, it is a suitable method to regard RelQSubject and RelQCategory as original features. For extended data, they are no original features.

For subtask B, we choose RelQbody and RelQ-Subject as the original features.

Observed Features The second kind of feature is obtained from data observation in subtask A. In *Factual* data, there are a lot of questions about Visas for couples, working, pets and animals, opportunities, etc. In *Opinion* data, questions that ask for advice are more common. In *Socializing* data, the questions are more colloquial, so it may include a word like *qler*. We chose the observed characteristic words as features. There are several examples for each class:

- *Factual*: *visit, license, husband, wife, embassy, sponsor*
- *Opinion*: *advice, school, suggestion, advise*
- *Socializing*: *ql, qler, weekend, love, going, today*

2.3 Model

Input Transformation Input sequence contains three special characters $Start$, $Extract$ and $Delim$, representing the start, end, and delimiter token respectively. We treat subtask A and B as question classification tasks. Their input formats are as follows.

- Subtask A: The most useful information in the data is the complete problem description RelQBody. We choose it as the main text and RelQSubject and RelQCategory as the original features. We employ the main text and two original features as the input. The observed features are also added to the input. The final input representation is $Start + RelQBody + Delim + RelQSubject + Delim + RelQCategory + Features + Extract$.
- Subtask B: We use RelCText as the main text, and use RelQBody and RelQSubject as original features. These features constitute the model input. We do not employ observed features in subtask B. So the final input format is $Start + RelQBody + Delim + RelCText + Delim + RelQSubject + Extract$

Model Description GPT is a language model, pre-trained on BooksCorpus (Zhu et al., 2015). There are 12-layer *Transformer* blocks. When optimizing, the training loss is the sum of text prediction loss and classification loss. Different tasks correspond to different input formats when using the GPT model for fine-tuning. Subtasks A and B input separately to the model. Figure 3 shows the input formats in detail.

3 Experiments

We present the experiments we conduct on our system and make a detailed analysis. We compare the performance between GPT and other models in subtask B. Follow Radford et al. (2018), we use the default model configuration for our model.

3.1 Results

We evaluate the systems on the development set and use accuracy as the main evaluation criteria. We use the organizer's score on the practice leaderboard of CodaLab as the baseline. Table 2 shows our performances on subtask A in detail. Original F and Observed F represent original features and observed features respectively. DE means Data Extension which we mentioned in section 2.1. Our best system achieves 81.59% in the development set.

	Acc	F1	AvgRec
GPT	0.7768	0.6392	0.6392
Above+Original F	0.7964	0.6738	0.6721
Above+Observed F	0.7992	0.6795	0.6771
Above+DE	**0.8159**	0.6959	0.6859

Table 2: Development result of subtask A. Acc means accuracy. Above means that a new change is added to the system which mentioned in previous row.

Table 3 shows our performances on subtask B. **F** represent features in the original dataset. We get 69.05% in the development set.

	Acc	F1	AvgRec	MAP
GPT	0.6369	0.4207	0.4312	0.7889
GPT+F	**0.6905**	0.4848	0.4789	0.7500

Table 3: Development result of subtask B. Acc means accuracy.

Using our best system we evaluate in the test set. As Table 4 shows, the score of our official submission is 81.95% in subtask A, which ranks sixth in all participants. The baseline of the test set is 45.0% in accuracy which lower than all the participants' score. In subtask B, we achieve 61.08% in the test set, which ranks seventh in all participants. The baseline of subtask B is 83.0%.

	Subtask A	Subtask B
Baseline	0.450	**0.830**
Our System	**0.8195**	0.6108

Table 4: Official submissions results on the test set for our system and the organizers's baselines. The metric is accuracy.

3.2 Analysis

Subtask A Adding original features proves to be useful to GPT, which increases by 2% than single GPT in accuracy. Observed features are not as useful as original features are. They only improve the result slightly. Data extension is also helpful, which improves the score by 1.67%.

Subtask B Original features are helpful for GPT and increase accuracy by 5.36%. It is a great improvement. The possible explanation might be that original features provide key information to the classification task in subtask B. However, their performance on the test set is not satisfactory, which only achieves 61.08%.

We implement the ESIM model (Chen et al., 2016) in subtask B, which applies bidirectional Long Short Term Memory (Hochreiter and Schmidhuber, 1997) and an alignment mechanism, achieving 63.69% in accuracy of the development set. Furthermore, we concatenate glove embeddings with contextual embeddings produced by ELMo (Peters et al., 2018) as features, improving accuracy of the development set by 2% in subtask B. Both results in subtask B are less than the best result of GPT, which is 69.05%. So we use GPT as our official system in subtask B.

4 Conclusions

We use the GPT model to participate in SemEval 2019 task 8. The goal of this task is question classification and answer classification. We demonstrate that large gains on fact checking can be realized by data extension, feature extraction, and input formats transformation. Our official submission achieves accuracy 81.95% of subtask A and 61.08% of subtask B, which ranks us 6th and 7th in the competition. What's more, features and data expansion are both helpful to the system.

For future work, we think data extension may be useful in subtask B since it performs well in subtask A. Furthermore, we would like to use external information in this task.

Acknowledgments

This work is funded by Beijing Advanced Innovation for Language Resources of BLCU TYR17001J the Fundamental Research Funds for the Central Universities in BLCU (No.17PT05) and the BLCU Academic Talents Support Program for the Young and Middle-Aged.

References

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2016. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*.

Naman Goyal. 2017. Learningtoquestion at semeval 2017 task 3: Ranking similar questions by learning to rank using rich features. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 310–314.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Qi Le, Zhang Yu, and Ting Liu. 2017. Scir-qa at semeval-2017 task 3: Cnn model based on similar and dissimilar information between keywords for question similarity. In *International Workshop on Semantic Evaluation*.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. *arXiv preprint arXiv:1710.03957*.

Tsvetomila Mihaylova, Georgi Karadzhov, Atanasova Pepa, Ramy Baly, Mitra Mohtarami, and Preslav Nakov. 2019. SemEval-2019 task 8: Fact checking in community question answering forums. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '19, Minneapolis, MN, USA.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. Semeval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 27–48.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Filip Šaina, Toni Kukurin, Lukrecija Puljić, Mladen Karan, and Jan Šnajder. 2017. Takelab-qa at semeval-2017 task 3: Classification experiments for answer retrieval in community qa. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 339–343.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Yufei Xie, Maoquan Wang, Jing Ma, Jian Jiang, and Zhao Lu. 2017. Eica team at semeval-2017 task 3: Semantic and metadata-based features for community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 292–298.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books.

CodeForTheChange at SemEval-2019 Task 8: Skip-Thoughts for Fact Checking in Community Question Answering

Adithya Avvaru[1,3] and **Anupam Pandey**[2,3]
[1]Teradata India Pvt. Ltd, India
[2]Qubole, India
[3] International Institute of Information Technology, Hyderabad, India
{adithya.avvaru,anupam.pandey}@students.iiit.ac.in

Abstract

Community Question Answering (cQA) is one of the popular Natural Language Processing (NLP) problems being targeted by researchers across the globe. Couple of the unanswered questions in the domain of cQA are 'can we label the questions/answers as factual or not?' and 'Is the given answer by the user to a particular factual question is correct and if it is correct, can we measure the correctness and factuality of the given answer?'. We have participated in SemEval-2019 Task 8 which deals with these questions. In this paper, we present the features used, approaches followed for feature engineering, models experimented with and finally the results. Our primary submission with accuracy (official metric for SemEval Task 8) of 0.65 in Subtask B (Answer Classification) and 0.63 in Subtask A (Question Classification) stood at 6^{th} and 16^{th} places respectively.

1 Introduction

Community Question Answering (cQA) forums such as Quora, StackOverflow, Yahoo! Answers, Qatar Living etc., now-a-days are fast and effective means of getting answers for any question. But the answers may or may not be correct and factual always. The focus of cQA research, for the last few couple of years, is revolving around determining the model which predicts the best answer for the question, given a question and a number of answers (might be hundreds or even thousands in number).

cQA is one of the popular problems being constantly in focus of SemEval organizers since 2015. The subtasks that were targeted earlier include (i) classifying the answer to a particular question as good or potentially good or bad in 2015[1], (ii) three reranking subtasks i.e., Question-Comment Similarity, Question-Question Similarity and Question-External Comment Similarity in 2016[2] and (iii) Question Similarity (QS) to detect duplicate questions and Relevance Classification (RC) in 2017[3]. Contrary to earlier tasks of SemEval focusing mainly on classification and similarity of questions and/or answers and/or comments, SemEval-2019 targets the factuality of the questions (whether the question is factual or not) and the factuality of the answers (whether the answers provided to the factual questions are factual or not). The tasks become more challenging as data have noisy (like *!!!*), and unstructured (like *Oh..*) words.

SemEval-2019 Task 8 features the following two subtasks:

Subtask A (Question Classification) - determine whether a question asks for a factual information, an opinion/advice or is just socializing. Example from the "Qatar Living" forum given in competition page[4] for this subtask is as follows:
Q: I have heard its not possible to extend visit visa more than 6 months? Can U please answer me.. Thankzzz...
answer 1: Maximum period is 9 Months....
answer 2: 6 months maximum
answer 3: This has been answered in QL so many times. Please do search for information regarding this. BTW answer is 6 months.
This subtask aims at building models to detect true factual information in cQA forums.

Subtask B (Answer Classification) - determine whether an answer to a factual question is true, false, or does not constitute a proper answer.
This subtask aims at building models that classify the answers into the following three categories, given a factual question: ***a)*** *Fac-*

[1]http://alt.qcri.org/semeval2015/task3/

[2]http://alt.qcri.org/semeval2016/task3/
[3]http://alt.qcri.org/semeval2017/task3/
[4]https://competitions.codalab.org/competitions/20022

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 1138–1143
Minneapolis, Minnesota, USA, June 6–7, 2019. ©2019 Association for Computational Linguistics

tual - True ***b)*** *Factual - False and* ***c)*** *Non-Factual.* The examples for each of them are as follows:

- **Factual - True:**
  Q: I wanted to know if there were any specific shots and vaccinations I should get before coming over [to Doha].
  A: Yes there are; though it varies depending on which country you come from. In the UK; the doctor has a list of all countries and the vaccinations needed for each.
- **Factual - False:**
  Q: Can I bring my pitbulls to Qatar?
  A: Yes you can bring it but be careful this kind of dog is very dangerous.
- **Non-Factual:**
  Q: Which is suggested - buy a new car or an used one?
  A: Its better to buy a new one.

We participated in both the subtasks of SemEval-2019 Task 8. For detailed description of the task, different approaches used by other participants and results obtained by all the participants, please refer the task description paper (Mihaylova et al., 2019).

The rest of the paper is organized as follows: Section 2 describes the related work. Section 3 describes the data used for this SemEval task. Sections 4 and 5 elucidate the system architecture (feature extraction and model building) and experimentation details (along with the results) respectively. Section 6 concludes the paper with focus on future research on this task.

2 Related Work

Some of the earlier works on cQA include the use of classification models - Support Vector Machines(SVMs) (Šaina et al., 2017; Nandi et al., 2017; Xie et al., 2017; Mihaylova et al., 2016; Wang and Poupart, 2016; Balchev et al., 2016) for Similarity tasks; Convolutional Neural Networks (CNNs) for Similarity tasks (Šaina et al., 2017; Mohtarami et al., 2016) and for answer selection (Zhang et al., 2017); Long-Short Term Memory (LSTM) model for answer selection (Zhang et al., 2017; Feng et al., 2017; Mohtarami et al., 2016); Random Forests (Wang and Poupart, 2016); LDA topic language model to match the questions at both the term level and topic level (Zhang et al., 2014); translation based retrieval models (Jeon et al., 2005; Zhou et al., 2011); XgBoost (Feng et al., 2017) and Feedforward Neural Network (NN) (Wang and Poupart, 2016).

All of the above related works on cQA used the features such as Bag of Words (BoW) (Franco-Salvador et al., 2016), Bag of vectors (BoV) (Mohtarami et al., 2016), Lexical features (for example, Cosine Similarity, Word Overlap, Noun Overlap, N-gram Overlap, Longest Common Substring/Subsequence, Keyword and Named Entity features etc.) (Franco-Salvador et al., 2016; Mohtarami et al., 2016; Nandi et al., 2017); Semantic features (for eg, Distributed representations of text, Knowledge Graphs, Distributed word alignments, Word Cluster Similarity, etc.) (Franco-Salvador et al., 2016); Word Embedding Features (like Word2vec[5] (Mikolov et al., 2013), GloVe[6](Pennington et al., 2014) etc.) (Wang and Poupart, 2016; Mohtarami et al., 2016; Nandi et al., 2017); Metadata-based features (like user information, answer length, question length, question marks in answer, question to comment length etc.) (Mohtarami et al., 2016; Mihaylova et al., 2016; Xie et al., 2017).

Another related task to cQA is Fact Checking in Community Forums (Mihaylova et al., 2018). This work doesn't involve classification of questions/answers based on factuality but it determines the veracity of the answer given a particular question. This work is related to our task in a way that the data being used in our task is annotated and released to the research community by Tsvetomila Mihaylova and her team.

The fact that this research problem is relatively new, the strengths of the scalable gradient tree boosting algorithm, XGBoost (Chen and Guestrin, 2016) and distributed sentence encoder, Skip-Thought vectors (Kiros et al., 2015) are not explored yet. We tried to apply and combine these two effective methods for finding factual nature of the questions and answers.

3 Data Description

The data for both Question Classification - Subtask A and Answer Classification - Subtask B, is organized into train, dev and test sets. The number of samples in each of these datasets is shown in the Table 1.

[5]https://code.google.com/archive/p/word2vec/
[6]http://nlp.stanford.edu/projects/glove/

Subtask	Datasets		
	Train	Dev	Test
A	1118	239	935
B	495	112	310

Table 1: Dataset Description

The data, in Question Classification, has both subject and body for each question. Similarly, for Answer Classification, the data has question subject, question body and an answer (as a comment text). The data of both the subtasks also have other information related to meta-data like user information, date and time of the question and answer post. The detailed description of data can be seen in task description paper (Mihaylova et al., 2019).

4 System Description

4.1 Feature Extraction

4.1.1 Data pre-processing

We have applied some basic preprocessing tasks like removing URLs, converting text to lowercase along with removing stopwords.

4.1.2 Extract Skip-Thought vectors

We choose Skip-Thought Vectors as word embeddings for this task mainly because these are highly generic sentence representations unlike GloVe or Word2Vec which averages word embeddings of each individual word to calculate the word embedding for a complete sentence.

In subtask A, we have retrieved Skip-Thought vectors for question body and question subject. In subtask B, we extracted Skip-Thought vectors for question body, question subject and answer comment. For both the subtasks, we have used the code[7] written by the Skip-Thought vectors' authors.

4.2 Model Building

Once we have extracted Skip-Thought vectors, we used these vectors to train different models - AdaBoost Classifier (only in case of Subtask B), DecisionTree Classifier, RandomForest Classifier, ExtraTrees Classifier, XGBoost Classifier and Multi-layer Neural Network with dropout layers in between, Adam optimizer and softmax activation in the final layer. The hyper-parameters of all the models is determined by applying Grid-Search with 10-fold cross-validation. The hyper-parameters are shown in the Table 2.

Classifier	Hyper-parameters
Decision Tree	min_samples_split = 2

Random Forest	n_estimators = 25
	min_samples_leaf = 1
	min_samples_split = 2

Extra Trees	n_estimators = 20
	max_features = 37

XGBoost	learning_rate = 0.1
	n_estimators = 100
	max_depth = 5
	objective = 'multi:softprob'

Adaboost	n_estimators = 45
	learning_rate = 1.0

Table 2: Hyper-parameters used for models

5 Evaluation and Results

5.1 Subtask A (Question Classification)

For this subtask, we extract Skip-Thought vectors as described in section 4.1.2. Once we get these two vectors, we generated four different combinations of vectors - (i) question body only, (ii) question subject only, (iii) concatenation vector of both question body and question subject and (iv) average vector of both question body and question subject. We trained all the models mentioned in the section 4.2 with each one of these vectors. The evaluation scores for these models on test data are shown in the Table 3.

5.2 Subtask B (Answer Classification)

For this subtask, we extract Skip-Thought vectors as described in section 4.1.2. Once we get these three vectors, we generated two different combinations of vectors - (i) concatenation vector of question body, question subject & answer and (ii) average vector of question body, question subject & answer. We trained all the models mentioned in the section 4.2 using each one of these embedding vectors. The evaluation scores for these models (except MAP scores) on test data are shown in the Table 4.

In both the tables 3 and 4, the column **Vector** represents Skip-Thought vector combination type (whether it is body only (in case of Subtask A) or subject only (in case of Subtask A) or

[7] https://github.com/ryankiros/Skip-Thoughts

Model	Vector	Accuracy	F-score	Avgrec
Decision Tree	**Bodies**	0.5728	0.3550	0.3893
	Subjects	0.5567	0.3308	0.3626
	Avg	0.5966	0.3904	0.4277
	Concat	0.5691	0.3498	0.3909
Extra Trees	**Bodies**	0.5406	0.3015	0.4075
	Subjects	0.5329	0.2992	0.4002
	Avg	0.5315	0.2902	0.4015
	Concat	0.5509	0.3158	0.4180
Random Forest	**Bodies**	0.5476	0.3119	0.4161
	Subjects	0.5329	0.2971	0.3950
	Avg	0.5567	0.3275	0.4236
	Concat	0.5446	0.3153	0.4139
Neural Network	**Bodies**	0.6849	0.5118	0.5426
	Subjects	0.6338	0.4404	0.4677
	Avg	**0.6884**	**0.5228**	**0.5561**
	Concat	0.6740	0.5007	0.5405
XGBoost	**Bodies**	0.6268	0.4382	0.5194
	Subjects	0.5959	0.4032	0.4646
	Avg**	0.6366	0.4474	0.5195
	Concat*	0.6299	0.4416	0.5130

Table 3: Evaluation scores for Subtask A
$*$ - marks the scores of our primary submission
$**$ - marks the scores of our contrastive submission
Row in bold - post evaluation accuracy score (improved over actual submission)

concatenation of vectors of body, subject and answer/comment or average of vectors of body, subject and answer/comment). On dev data set, XGBoost Classifier with concatenated Skip-Thought vectors generated best scores for both subtasks. Hence, these are part of final submissions.

However, the rows which are marked in bold (in both subtasks) produced best accuracy score with Multi-layer Neural Network Classifier beating the best score of our CodaLab final submission. The Multi-layer Neural Network is designed to have an input layer, 2 hidden layers and an output layer with "relu" activations at input and hidden layers and "sigmoid" activation at output layer. All the layers are trained with 50 neurons except the output layer which has one neural node. This model counters overfitting problem by introduction of intermittent Dropout layers.

Another interesting observation that we found is the models, surprisingly, performed better when URLs are kept in the text compared to when URLs were removed.

Model	Vector	Accuracy	F-score	Avgrec
Decision Tree	**Avg**	0.5354	0.2755	0.3791
	Concat	0.5438	0.2843	0.3284
Extra Trees	**Avg**	0.5763	0.2845	0.3229
	Concat	0.6021	0.3150	0.3558
Random Forest	**Avg**	0.6215	0.3068	0.3285
	Concat	0.6172	0.2890	0.2943
Adaboost	**Avg**	0.5570	0.2607	0.2813
	Concat	0.5743	0.2612	0.2564
Neural Network	**Avg**	0.6129	0.3434	0.4036
	Concat	**0.6752**	**0.3420**	**0.3559**
XGBoost	**Avg****	0.6150	0.3225	0.3529
	Concat*	0.6537	0.3252	0.1555

Table 4: Evaluation scores for Subtask B
$*$ - marks the scores of our primary submission
$**$ - marks the scores of our contrastive submission
Row in bold - post evaluation accuracy score (improved over actual submission)

6 Conclusion

The earlier works on cQA didn't use Skip-Thought vectors, to the best of our knowledge. Hence, we used these vectors for both subtasks. We also have tried unique combinations of Skip-Thought vectors of question body, question subject and comments/answers (only in case of Subtask B) - either concatenation or average of vectors with different models. Out of all the models, concatenated Skip-Thought vectors with XGBoost Classifier generated best result out of all the combinations; as a result of which we stood 6^{th} in Subtask B and 16^{th} in Subtask A. However, post-evaluation submission which used concatenated Skip-Thought vectors with Neural Network classifier produced better accuracy score of 0.6752 compared to 0.6537 (which is official best result for Task B) and 0.6884 compared to 0.6299 (which is official best result for Task A). However, in future we would like to extend our work with other word embeddings like Word2vec, GloVe and BERT (Devlin et al., 2018) features and compare the results with current work using Skip-Thought vectors.

Acknowledgments

This research is supported by Teradata, India and Qubole, India in collaboration with Language Technologies Research Centre (LTRC) of International Institute of Information Technology, Hyderabad, India.

References

Daniel Balchev, Yasen Kiprov, Ivan Koychev, and Preslav Nakov. 2016. PMI-cool at SemEval-2016 Task 3: Experiments with PMI and Goodness Polarity Lexicons for Community Question Answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 844–850, San Diego, California. Association for Computational Linguistics.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 785–794. ACM.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Wenzheng Feng, Yu Wu, Wei Wu, Zhoujun Li, and Ming Zhou. 2017. Beihang-MSRA at SemEval-2017 Task 3: A Ranking System with Neural Matching Features for Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 280–286, Vancouver, Canada. Association for Computational Linguistics.

Marc Franco-Salvador, Sudipta Kar, Thamar Solorio, and Paolo Rosso. 2016. UH-PRHLT at SemEval-2016 Task 3: Combining Lexical and Semantic-based Features for Community Question Answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 814–821, San Diego, California. Association for Computational Linguistics.

Jiwoon Jeon, W Bruce Croft, and Joon Ho Lee. 2005. Finding Similar Questions in Large Question and Answer Archives. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 84–90. ACM.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-Thought Vectors. *arXiv preprint arXiv:1506.06726*.

Tsvetomila Mihaylova, Pepa Gencheva, Martin Boyanov, Ivana Yovcheva, Todor Mihaylov, Momchil Hardalov, Yasen Kiprov, Daniel Balchev, Ivan Koychev, Preslav Nakov, Ivelina Nikolova, and Galia Angelova. 2016. SUper Team at SemEval-2016 Task 3: Building a Feature-Rich System for Community Question Answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 836–843, San Diego, California. Association for Computational Linguistics.

Tsvetomila Mihaylova, Georgi Karadzhov, Atanasova Pepa, Ramy Baly, Mitra Mohtarami, and Preslav Nakov. 2019. SemEval-2019 task 8: Fact Checking in Community Question Answering Forums. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '19, Minneapolis, MN, USA.

Tsvetomila Mihaylova, Preslav Nakov, Lluis Marquez, Alberto Barron-Cedeno, Mitra Mohtarami, Georgi Karadzhov, and James Glass. 2018. Fact Checking in Community Forums. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mitra Mohtarami, Yonatan Belinkov, Wei-Ning Hsu, Yu Zhang, Tao Lei, Kfir Bar, Scott Cyphers, and Jim Glass. 2016. SLS at SemEval-2016 Task 3: Neural-based Approaches for Ranking in Community Question Answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 828–835, San Diego, California. Association for Computational Linguistics.

Titas Nandi, Chris Biemann, Seid Muhie Yimam, Deepak Gupta, Sarah Kohail, Asif Ekbal, and Pushpak Bhattacharyya. 2017. IIT-UHH at SemEval-2017 Task 3: Exploring Multiple Features for Community Question Answering and Implicit Dialogue Identification. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 90–97, Vancouver, Canada. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Filip Šaina, Toni Kukurin, Lukrecija Puljić, Mladen Karan, and Jan Šnajder. 2017. TakeLab-QA at SemEval-2017 Task 3: Classification Experiments for Answer Retrieval in Community QA. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 339–343, Vancouver, Canada. Association for Computational Linguistics.

Hujie Wang and Pascal Poupart. 2016. Overfitting at SemEval-2016 Task 3: Detecting Semantically Similar Questions in Community Question Answering Forums with Word Embeddings. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 861–865, San Diego, California. Association for Computational Linguistics.

Yufei Xie, Maoquan Wang, Jing Ma, Jian Jiang, and Zhao Lu. 2017. EICA Team at SemEval-2017 Task 3: Semantic and Metadata-based Features for Community Question Answering. In *Proceedings of the*

11th International Workshop on Semantic Evaluation (SemEval-2017), pages 292–298, Vancouver, Canada. Association for Computational Linguistics.

Kai Zhang, Wei Wu, Haocheng Wu, Zhoujun Li, and Ming Zhou. 2014. Question Retrieval with High Quality Answers in Community Question Answering. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 371–380. ACM.

Sheng Zhang, Jiajun Cheng, Hui Wang, Xin Zhang, Pei Li, and Zhaoyun Ding. 2017. FuRongWang at SemEval-2017 Task 3: Deep Neural Networks for Selecting Relevant Answers in Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 320–325, Vancouver, Canada. Association for Computational Linguistics.

Guangyou Zhou, Li Cai, Jun Zhao, and Kang Liu. 2011. Phrase-Based Translation Model for Question Retrieval in Community Question Answer Archives. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 653–662. Association for Computational Linguistics.

ColumbiaNLP at SemEval-2019 Task 8: The Answer is Language Model Fine-tuning

Tuhin Chakrabarty
Columbia University
Department Of Computer Science
tc2896@columbia.edu

Smaranda Muresan
Columbia University
Data Science Institute
smara@columbia.edu

Abstract

Community Question Answering forums are very popular nowadays, as they represent effective means for communities to share information around particular topics. But the information shared on these forums is often not correct or misleading. This paper presents the ColumbiaNLP submission for the SemEval-2019 Task 8: Fact-Checking in Community Question Answering Forums. We show how fine-tuning a language model on a large unannotated corpus of old threads from Qatar Living forum helps us to classify question types (factual, opinion, socializing) and to judge the factuality of answers on the shared task labeled data from the same forum. Our system finished 4th and 2nd on Subtask A (question type classification) and B (answer factuality prediction), respectively, based on the official metric of accuracy.

1 Introduction

Community Question Answering (cQA) forums such as StackOverflow, Yahoo! Answers, and Quora are very popular nowadays, as they represent effective means for communities to share information and to collectively satisfy their information needs. Questions asked on these sites can be of different types, and the answers can often be false, misleading or irrelevant.

SemEval-2019 Task 8 is structured around two subtasks. Subtask A is a question classification task, where the questions types are:

- **Factual**: The question is asking for factual information, which can be answered by checking various information sources, and it is not ambiguous (e.g., "What is Ooredoo customer service number?").
- **Opinion**: The question asks for an opinion or an advice, not for a fact. (e.g., "Can anyone recommend a good Vet in Doha?"")
- **Socializing**: Not a real question, but intended for socializing or for chatting. This can also mean expressing an opinion or sharing some information, without really asking anything of general interest (e.g., "What was your first car?")

Subtask B is an answer classification task: are the answers to *factual questions* factual or not, and if they are factual are they true or false:

- **Factual - TRUE**: The answer is True and can be proven with an external resource. (Q: "I wanted to know if there were any specific shots and vaccinations I should get before coming over [to Doha]."; A: "Yes there are; though it varies depending on which country you come from. In the UK; the doctor has a list of all countries and the vaccinations needed for each.").
- **Factual - FALSE**: The answer gives a factual response, but it is False, it is partially false or the responder is unsure about (Q:"Can I bring my pitbulls to Qatar?"; A: "Yes you can bring it but be careful this kind of dog is very dangerous.").
- **Non-Factual**: When the answer does not provide factual information to the question; it can be an opinion or an advice that cannot be verified (e.g., "It's better to buy a new one.").

2 Related Work

Yu and Hatzivassiloglou (2003) separated opinions from fact, at both the document and sentence level.

(Mihaylova et al., 2018) were the first to propose a novel multi-faceted model for fact checking of answers on community question answering forums. Their proposed model captures information

OPINION	FACTUAL	SOCIALIZING
586	311	254

Table 1: Size of Subtask A dataset (question types).

TRUE	FALSE	NON-FACTUAL
166	135	194

Table 2: Size of Subtask B dataset (answer types).

QUESTIONS	ANSWERS
189,941	1,894,456

Table 3: External unannoted questions and answers.

from the answer content (what is said and how), from the author profile (who says it), from the rest of the community forum (where it is said), and from external authoritative sources of information (external support). (Nakov et al., 2017) proposed models for credibility assessment in community question answering forums. However, credibility is different from veracity as it is a subjective perception about whether a statement is credible, rather than verifying whether it is true/false as a matter of fact.

Recently there has been a lot of attention on building models for fact checking. (Thorne et al., 2018) introduce a new publicly available dataset for fact extraction and verification (FEVER Shared Task). The dataset consists of 185,445 claims generated by altering sentences extracted from Wikipedia, and the task is to classify claims as SUPPORTED, REFUTED or NOTENOUGHINFO. However, the verification of the claims is limited to a particular database (namely Wikipedia) unlike Subtask B. Also, the claims are inherently less noisy as compared to answers in Community Question Answering forums.

Pre-trained language models have been recently used to achieve state-of-the-art results on a wide range of NLP tasks (e.g., sequence labeling and sentence classification). Some of the recent works that have employed pre-trained language models include (Howard and Ruder, 2018), (Peters et al., 2018), (Yang et al., 2018), and (Radford et al., 2018). In this paper, we show the effectiveness of the Universal Language Model Fine-tunig (ULMFiT) method (Howard and Ruder, 2018) for both question classification and answer fact checking.

3 Data

One of key challenges for both Subtask A and B is the limited amount of annotated data. This poses a challenge to apply state-of-the-art neural discrimination models without using additional data.

3.1 Labeled Data

Subtask A has a total of 1,118 questions divided into three types. Table 1 show the class distribution. Subtask B has a total of 495 answers divided into three types. Table 2 shows the class distribution.

3.2 Unlabeled Data

The task allows the use of external unnannoted data of 189,941 threads from Qatar Living Forum. Each of these threads have questions and answers just as our training data but without any labels. These threads may contain enough information to estimate the factuality of the answers in Subtask B as well as linguistic patterns in the questions asked for Subtask A. We refer to the resulting collection of comments as the **QL** dataset.

4 Model and Analysis

As the QL data is from the same distribution as our shared task data (Quatar Living), we need a method of incorporating this dataset into our models for both subtasks. We use a language model fine-tuning approach, which requires only unlabeled data similar to the task of interest.

The Universal Language Model Fine-Tuning method (ULMFiT) (Howard and Ruder, 2018) consists of the following stages: a) General-domain LM pre-training b) Task-specific LM fine-tuning and c) Task-specific classifier fine-tuning. In stage (a), the language model is trained on Wikitext-103 (Merity et al., 2017) consisting of 28,595 pre-processed Wikipedia articles and 103 million words capturing general properties of language. Stage (b) fine-tunes the LM on task-specific data, as no matter how diverse the general-domain data used for pre-training is, the data of the target task will likely come from a different distribution. In stage (c), a classifier is trained on the target task, fine-tuning the pre-trained LM but with an additional layer for class prediction. The models use a stacked Long Short Term Memory (LSTM) network to represent each sentence. For stages (a) and (b), the output of the LSTM is used to make a prediction of the next token and the parameters from stage (a) are used to initialize stage

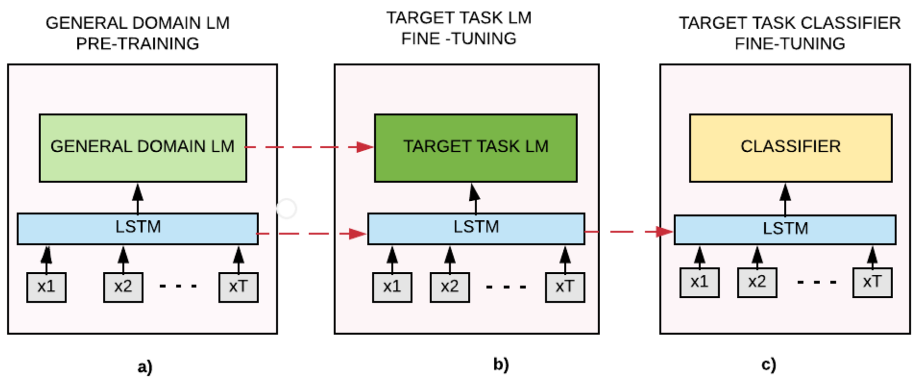

GENERAL DOMAIN LM PRE-TRAINING
TARGET TASK LM FINE -TUNING
TARGET TASK CLASSIFIER FINE-TUNING
GENERAL DOMAIN LM
TARGET TASK LM
CLASSIFIER
LSTM
x1
x2
xT
a)
b)
c)

Figure 1: Schematic of ULMFiT showing the three stages. The dashed arrows indicate that the parameters from the previous step were used to initialize the next step.

Task Specific LM Fine-Tuning	QL LM Fine-Tuning
65	**81**

Table 4: Accuracy on the test splits while doing cross validation on training data for Subtask A

(b). For stage (c), the model is initialized with the same LSTM but with a new classifier layer given the output of the LSTM.

This process is illustrated in Figure 1. We refer the reader to Howard and Ruder (2018) for further details. In our work, we maintain stages (a) and (c) but modify stage (b) so that we fine-tune the language model on the unlabelled data rather than the task-specific data. The goal of ULMFiT is to allow training on small datasets of only a few hundred examples, but our experiments will show that fine-tuning the language model on the QL data improves over only task-specific LM fine-tuning.

4.1 Subtask A

For Subtask A we fine-tune a language model on the 189,941 questions from the QL dataset. Our initial experiments show that fine-tuning the LM on the QL dataset give large performance gains over fine-tuning on task specific data as demonstrated in Table 4.

Lets take the following question:

> *Ramadan Working Hours? For companies who are operating 5 days a week; what are your timings? Ours is 8:00am to 3:00pm.?*

This is a **Factual** question, but the task-specific LM fine-tuning labels it as **Socializing**, while fine-tuning on QL data allows the model to correctly classify it as **Factual**. To understand why this happens, we delve deeper into the unlabeled data set where we find multiple similar questions based on TF-IDF similarity, demonstrating that the LM Fine-Tuning on QL data learns representations of questions based on discriminatory phrases.

- ***Ramadan Working Hours?*** *Eid holidays announced*
- ***Ramadan Working Hours?*** *good morning; Did anybody knows what is the right time* ***timing*** *or* ***working hours*** *during* ***Ramadan****? Thanks and advance.*
- ***Ramadan Working Hours?*** *Ministry of Civil Service Affairs and Housing has issued a circular in this regard defining the restricted* ***working hours****. Can somebody help me to find the English translation of that. Thank you*
- *help pls! can somebody tell me the* ***Ramadan Working Hours?*** *of ministry of Foreign Affairs???*

On the official test data, the ULMFiT approach where the target task classifier is fine-tuned on the LM fine-tuned on questions from QL data gives us an accuracy of **83** placing us 4th on the leaderboard.

4.2 Subtask B

For Subtask B we followed a similar approach of LM fine-tuning. We obtained representations of answers by fine-tuning a LM on 1,894,456 answers from the QL dataset. Next, we obtained

ANSWER	AVG COSINE SIMILARITY
Medical Check is for everyone mate	0.81
The test is done for everyone; but is restricted to the above categories u mentioned.	0.44
Regardless what your job is...everybody gets tested for hepatitis B/C cheers Never say never	0.76
As I told u hepatitis B/C are checked for everyone applying for residence permit.If the result is positive u go back where u came from. And I know all of the above because my husband is a consultant pathologist. cheers	0.72

Table 5: Average Cosine Similarity scores of contextual representations of each answer to every other answer in the thread

representations by fine-tuning a LM on 1,894,456 question-answer pairs, in order to capture whether an answer is actually suited for the question asked or something irrelevant. An answer which is relevant to the question asked can then be easily discerned from an irrelevant one by a discriminative classifier.

Our model did not take into account external evidence from search engines as done by (Mihaylova et al., 2018), so we had to rely on intra-forum evidence for factuality features. Our hypothesis is that for factual questions, the answers which are factually true are similar to each other, while answers which are false or irrelevant are different from other answers. We incorporated this behaviour in our model: for every answer we computed the cosine similarity between the contextual representation of the answer obtained from last layer of the LSTM used to train the language model for answers. For each answer, we averaged the cosine similarity between that answer and the other answers in the same thread.

For example, take the question:

> *Hi all; are hepatitis B and C checked for in the medical test for non-medical professionals? Basically;I have been getting conflicting information on this. Some say that Hep B and C are tested for everyone applying for residence permit. Others say that only medical professionals; primary school teachers and food handlers are tested for Hep B and C. Please discuss!*

From Table 5 we see that the answer with the lowest cosine similarity (0.44) is the answer which is factually false, compared to the other answers which are factually true and have a higher cosine similarity.

We use these answer representations, question-answer pair representations and the average cosine similarity as features to train an XGB classifier to obtain our final results. For threads with only 1 answer we took the cosine similarity as 0.5.

FEATURES	ACCUARCY
LM on answers	61
LM on Q-A pairs	64
LM on answers and Q-A pairs	72
LM on answers and Q-A pairs + Avg Cosine Similarity	**79**

Table 6: Ablation scores for Subtask B on the final test set for Task 8.

Table 6 shows us the ablation scores for each approach. The best accuracy obtained is a combination of all the approaches together. We obtain an accuracy of **79** placing us 2nd on the leaderboard. The absence of true labels for both the dev and the test set prevents us from conducting an error analysis.

5 Implementation

The language model is fine tuned for 15 epochs as done in the ULMFIT original paper for both Subtasks. For classifier fine-tuning we use the same hyper-parameters as (Howard and Ruder, 2018) except the learning rate which is set to .0001 . We train our classifier for 5 epochs on both sub-tasks. Each model was run 10 times to account for variance and the results reported for both the tasks are the average of 10 runs. We did not use any special pre-processing technique and use the same approach as done in the ULMFIT paper, i.e clean up extra spaces, tab characters, new line characters and other characters and replace them with standard ones. We also use Spacy library to tokenize the data. The implementation can be found here [1]

6 Conclusion

We show that fine-tuning a language model on a large unsupervised corpus from the same community forum helps us achieve better accuracy for question classification. Most community

[1]https://github.com/fastai/fastai/blob/master/courses/dl2/imdb.ipynb

question-answering forums have such unlabeled data, which can be used in the absence of large labeled training data .

For answer classification we show how we can leverage information from previously answered questions on the thread through language model fine tuning. Our experiments also show that modeling an answer individually is not the best idea for fact-verification and results are improved when considering the context of the question.

Determining factuality of answers definitely requires modeling world knowledge or external evidence. The questions asked are often very noisy and require reformulation. As a future step we would want to incorporate external evidence from the internet in the factual answer classification problem.

References

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers)*, pages 328–339.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. In proceedings of the international conference on learning representations.

Tsvetomila Mihaylova, Preslav Nakov, Llu'is Marquez, Alberto Barron-Cede'no, Mitra Mohtarami, Georgi Karadzhov, and James Glass. 2018. Fact checking in community forums. In *Association for the Advancement of Artificial Intelligence*.

Preslav Nakov, Tsvetomila Mihaylova, Llu'is Marquez, Y Shiroya, and I Koychev. 2017. Do not trust the trolls: Predicting credibility in community question answering forums. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, pages 551–560.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of NAACL-HLT 2018.*, pages 809–819.

Zhilin Yang, Jake Zhao, Bhuwan Dhingra, Kaiming He, William W. Cohen, Ruslan Salakhutdinov, and Yann LeCun. 2018. Glomo: Unsupervisedly learned relational graphs as transferable. In *arXiv:1806.05662*.

Hong Yu and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*.

DOMLIN at SemEval-2019 Task 8: Automated Fact Checking exploiting Ratings in Community Question Answering Forums

Dominik Stammbach
DFKI, Saarbrücken, Germany
dominik.stammbach@dfki.de

Stalin Varanasi
DFKI, Saarbrücken, Germany
stalin.varanasi@dfki.de

Günter Neumann
DFKI, Saarbrücken, Germany
neumann@dfki.de

Abstract

In the following, we describe our system developed for the Semeval2019 Task 8. We fine-tuned a BERT checkpoint on the qatar living forum dump and used this checkpoint to train a number of models. Our hand-in for subtask A consists of a fine-tuned classifier from this BERT checkpoint. For subtask B, we first have a classifier deciding whether a comment is factual or non-factual. If it is factual, we retrieve intra-forum evidence and using this evidence, have a classifier deciding the comment's veracity. We trained this classifier on ratings which we crawled from qatarliving.com.

1 Introduction

This paper contains our system description for the SemEval2019 task 8 about Fact Checking in Community Forums. The task 8 is divided into two subtasks: In subtask A, the goal is to determine whether a question asks for a factual answer, an opinion or is just posed to socialize. In subtask B, if we have a question asking for a factual answer, we classify the answers to such a question into three categories, namely the answer is either true, false or non-factual, i.e. it does not answer the question in a factual way.

For subtask A, we trained a BERT classifier on the training set and optimized hyper-parameters on the development set. For subtask B, we decided to tackle the challenge with two binary classifiers: Firstly, we decide whether a comment is factual or not. If our classifier decides that a comment is factual, we retrieve intra-forum evidence to determine the comment's veracity using a textual entailment approach. Given the small training set for subtask B, we decided to leverage openly available information on qatarliving.com to create a medium-sized training set. We found that comments on qatarliving.com are sometimes associated with ratings[1] (ranging from 1 to 5) and discovered that high ratings often correspond to replies answering the question in a true way. If a comment has recieved a low rating, we inferred that the comment was most likely not helpful to answer the question and therefore we decided to treat it as a false reply.

2 Related Work

Automated Fact Checking is recently mostly perceived as a number of tasks which can be pipelined together. In the FEVER shared task, most participating systems would first find evidence and then train textual entailment models (Thorne et al., 2018). Related work for Fact Checking in community forums considers a multi-faceted approach incorporating firstly what is said, how it is said and by whom and secondly external evidence from either the web or from the forum itself (Mihaylova et al., 2018). An SVM is trained on top of these features to decide the veracity of a comment.

In our system, we took a similar approach by first retrieving possible evidence, secondly filtering such evidence (through another classifier) and eventually train a system which decides the veracity of a comment based on whether the comment is entailed by the found evidence or not.

3 System Description

Recent progress in natural language understanding shows that pre-training transformer decoders on language modelling tasks leads to remarkable transferable knowledge which boosts performance on a wide range of NLP tasks (Radford et al., 2018). The most recent development then is the

[1]We learnt after the deadline of the shared task that these ratings were automatically generated: https://www.qatarliving.com/forum/technology-internet/posts/searching-information-qatar-living-has-just-grown-faster

Deep Bidirectional Transformers (BERT) which is jointly pre-trained on a masked language modelling task (therefore bidirectional) and on a next-sentence prediction task pushing already impressive results even further (Devlin et al., 2018). All our classifiers in our hand-in are fine-tuned BERT models.

3.1 Domain Adaptation

We firstly fine-tuned a BERT checkpoint (pre-trained on uncased English data only) on the unannotated dataset from Qatar Living with 189,941 questions and 1,894,456 comments (Nakov et al., 2016). Fine-tuning a BERT checkpoint on a new domain consists of further training it jointly on the masked language modelling task and the next-sentence prediction task. For this dataset, it is not always trivial to decide what a sentence is and we use whole comments later on anyways, so we replaced the next-sentence prediction task by a next-comment prediction task, that is our model has to guess whether two comments are appearing consecutively in a thread or not.

Given the peculiarities of the BERT tokenizer, we cleaned the dataset through the following steps:

- we lowercased all characters
- we replaced a character which appears more than three times consecutively to only appear three times ("!!!!!!!!!" then becomes "!!!")
- we removed user specific quotes
- we removed comments containing a type/token ratio[2]of less than 0.15 (because we noticed that they are mostly spam)
- we replaced urls with a special token "url", phone numbers with a special token "tel" and email addresses with a special token "email"

In Table 1, we show the masked language modelling accuracy (MLM) and next-comment prediction accuracy (NC) for the uncleaned and the cleaned version, both fine-tuned for 100k steps. We also show results for training a task-specific model for subtask A (accuracy on the development set) with the stand-alone BERT model, a fine-tuned model on the raw data and a fine-tuned model on the cleaned data.

[2]https://en.wikipedia.org/wiki/Lexical_density

System	MLM	NC	task A
not fine-tuned	-	-	0.80
fine-tuned raw data	0.68	1	0.79
fine-tuned cleaned data	0.57	0.89	**0.84**

Table 1: Effect of cleaning the dataset

We capped characters to only appear maximum three times consecutively. If they appear more often, they would form a subword anyways and we think it is too easy for the model to guess such subwords in longer sequences (consider the sequence "!!!!<MASKED>!!!!!"). Users in the forum can add specific quotes which are appended to their posts, e.g. one user chose the ending *life's too short so make the most of it; you only live but once...* which appears 3865 times in the data. We refer to this as "user specific quotes" and removed them as we believe the model would overfit on such quotes during fine-tuning and would not learn useful knowledge about the domain while doing so. Lastly, we believe that there is not much value to be gained in learning urls, phone numbers and emails, and they often get splitted into a long series of subword units (the vocabulary is managed through byte-pair encoding). We think, these reasons combined make the model learn such patterns very well (resulting in a higher accuracy for the BERT tasks for the model trained on the raw data), but it does not gain much transferable knowledge by doing so, resulting in a lower accuracy for subtask A.

3.2 Subtask A

For subtask A, we trained a task-specific BERT classifier from the fine-tuned BERT checkpoint explained above. Fine-tuning such a classifier consists of learning embeddings for a special classification token, let the model compute self-attention over its 12 layers and finally gather the hidden representation of the classification token (the first token in the sequence usually). This hidden representation is fed into one hidden layer and lastly one classification layer. The input to the model is the concatenation of the question's subject and its body and we regularize the model by applying a dropout of 0.1 on the classification layer. We grid-searched over the proposed hyper-parameter range in the BERT paper (that is initial learning rate, batch-size and number of fine-tuning epochs) (Devlin et al., 2018).

In Table 2, we report the accuracy on the development set for a number of experiments with different features. RelQBody (the opening post by the thread creator) is the question's body, RelQSubject the question's subject (the title of a thread) and RelQ_Category its category (the name of the sub-board it has been posted in). We concatenated the different features with whitespaces in between.

Feature	acc
RelQBody	0.82
RelQSubject + RelQBody	**0.84**
RelQ_Category + RelQSubject + RelQBody	0.83

Table 2: Accuracy for different features for subtask A

Using only the question's body results in slightly worse results than the concatenated subject and body. We also tried to add the category, that is the name of the sub-forum a question has be posted in. The rationale here is that one subboard on qatarliving is called "Socialising" and we thought it might give the model a cue that questions there are more prone to be of the class socializing. However, we get slightly worse results by including it. Our final hand-in eventually consists of an ensemble of 5 models (the voting strategy is majority voting) which are trained on the concatenation of the subject and the body of a question.

Our system ranked fifth with an accuracy of 82% on the test set.

3.3 Subtask B: Overview

As we described earlier, we decided to tackle subtask B as a series of different tasks and for each, we trained different models:

1. decide whether a comment is factual or non-factual
2. retrieve related threads (based on the question of a thread)
3. filter for relevant comments in related threads
4. train a textual entailment[3] system, that is whether the evidence entails a claim or not

For the first step, we have fine-tuned a BERT checkpoint on the SQuAD question answering corpus (Rajpurkar et al., 2016). If a comment contains the answer to a question, we consider it as factual and have to check its veracity in a further step. If the answer to a question can not be found in the comment, we label it as non-factual. If the answer can be found in a comment, i.e. we have a factual comment, we continue with steps 2-4.

For the second step, we search for intra-forum evidence in the qatar living forum dump (Nakov et al., 2016). We concatenate the subject and body of each thread. We lowercase all the tokens, remove all characters except the letters a-z and use the snowball stemmer (Porter, 2001) for stemming the tokens. Afterwards, we search for the most similar threads using TF-IDF[4] and keep the five most similar threads.

We also manually evaluated whether gigablast[5] and the duckduckgo API[6] would yield useful evidence, but after having checked 15 sampled questions from the development set manually, we decided to not pursue this any further. First of all, if we just use the question's subject concatenated with its body as the query for the search engine, it would not be precise and most such queries would not return relevant web pages. One has to summarize this large text of the question automatically into a query suitable for a web search engine. We manually created search-engine searchable queries for the 15 sampled questions and found that only two of such queries returned relevant results. This may be because there is less information available on the internet for queries regarding living in Qatar except for the forum qatarliving.com itself. Hence, we decided to let go of the idea of using publicly available web search engines with automatically summarized questions for this task.

For the third step, we trained a BERT model on the concatenation of the SemEval2016 task 3 subtask A and subtask C data to filter the intra-forum evidence. The input to the model is the original question (the one we want to fact-check comments for) and the found replies in the most similar threads. The output is whether a comment answers that question in a relevant way (yes or no). For the test set for task B, we found 642 comments via the TF-IDF search engine and after filtering the comments, we are left with 162 comments as evidence (24% of these 642 comments).

For the fourth and last step, we also used a BERT model. This model should predict the veracity of a comment given the retrieved evidence in step two and three. However, given the small

[3]https://en.wikipedia.org/wiki/Textual_entailment

[4]https://radimrehurek.com/gensim/

[5]https://www.gigablast.com/

[6]https://duckduckgo.com/api

size of the training set for subtask B (135 false and 166 true comments), we did not manage to find a suitable hyper-parameter configuration which would yield a model with decent performance on the development set.

3.4 Subtask B: Textual Entailment Model

While looking at the forum online, we noticed that some comments in the forum are associated with ratings (Figure 1). Such ratings can range from 1 to 5 and we found that comments with a rating of 5 tend to answer questions in a true way and comments with a rating of 2 or 3 tend to have not been that helpful (we did not find any comments with a rating of 1).

Hence, we have crawled the threads from the forum dump (Nakov et al., 2016) online so that we get the corresponding ratings. We found that the url of a thread is a combination of the sub-forum a thread has been posted in and its subject (with whitespaces replaced with a ”+” and some stop-words removed) and reverse engineered the name of the urls. We ignored the threads for which we couldn’t find the corresponding web page automatically. After having crawled the website for one night (with short pauses after each call to the website), we ended up with 19’000 comments with a rating of 5 and 13’000 comments with a rating of 2 or 3, resulting in a corpus with 32’000 examples. With this corpus, we trained a textual entailment system which predicts whether a comment is associated with a rating of 2-3 or 5 (we left out comments with a rating of 4 and comments without a rating).

We then retrieved intra-forum evidence as described above for all these 32’000 comments and trained our BERT checkpoint (which was pre-trained on the forum dump) on that corpus and obtain ”question-comment-evidence” triplets. Let us assume the question is *”Where can I get Potassium Nitrate?”*, the comment is *”Try Metco industrial area. 465 1234”* and we retrieve two evidence texts *”potassium nitrate are not allowed to buy here in qatar. you have to ask a permission from the police department or to the civil defense...”* and *”not sure if same as what you want; but i got potassium before from pharmacies...”*. We then form two triplets (one for each evidence text) and let the model predict an output for each.

Since the different retrieved evidence for each claim is independent, we thought that it would be a bad idea to just concatenate all the evidence and use that as input to our classifier. We therefore decided to aggregate the outputs of each triple using the logsumexp function (Eq. 1) which is a smooth version of the max function and allows the model to back-propagate dense gradients (Verga et al., 2018). We think this lets the model also figure out on its own which evidence it should look out for.

$$scores(i) = log \sum exp(A_{ij}) \quad (1)$$

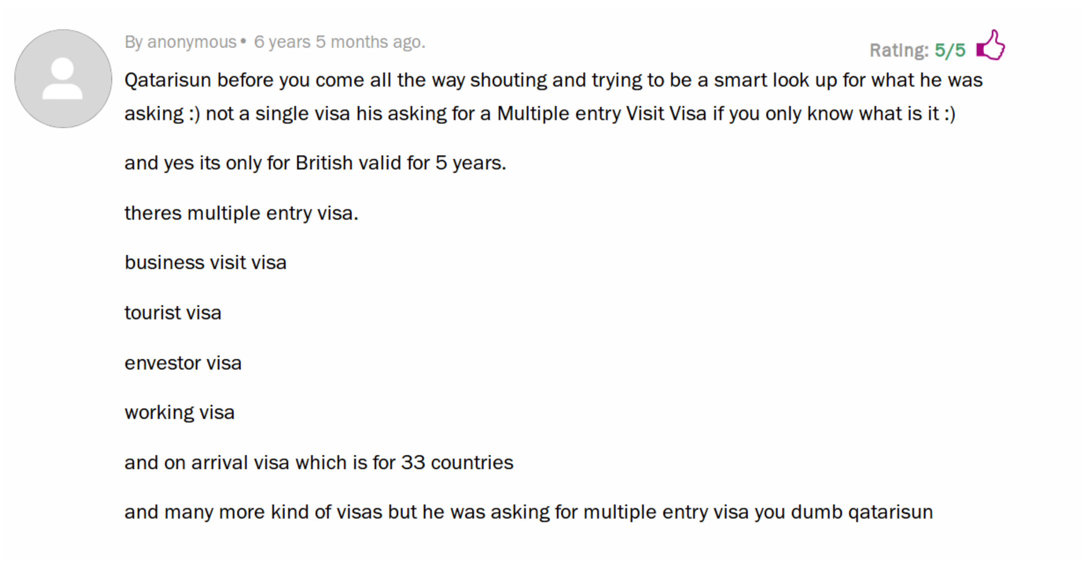
A is a matrix with two columns (bad rating or good rating) in which we stack the predictions for each ”question-comment-evidence” triplet. That is, each row in that matrix is the prediction for a comment with a rating given one evidence comment found in the forum. In comparison to the normal max function (which back-propagates sparse gradients), we learn from each comment-evidence pair and not only from the one with the highest scores.

We trained that model with a batch-size of 8 answers and for each answer, we retrieve 4 evidence comments (resulting in 32 triplets). During test time, we retrieve up to 8 evidence comments, predict results for each triplet and aggregate the predictions for each triplet using the logsumexp function to yield a final classification. In Table 3, we show the results of our two classifiers on the training set of subtask B (because we did not use that set for training at all).

class	pr	rc	F1
non-factual	0.43	0.53	0.47
factual	0.63	0.53	0.58
factual false	0.36	0.52	0.42
factual true	0.71	0.56	0.62

Table 3: Results on training set of subtask B

The first two rows show the results of our BERT model trained on the SQuAD corpus. The factual class contains the examples which are true or false. After having performed a manual error analysis for the factual and non-factual class, we conclude that we disagree with some of the annotations in the training corpus. The last two rows show the performance of our classifier trained on ratings on the training set. For the true answers, it performs better than for the false answers (which might be due to a slight imbalance of training examples in our compiled corpus).


By anonymous • 6 years 5 months ago.

Rating: 5/5

Qatarisun before you come all the way shouting and trying to be a smart look up for what he was asking :) not a single visa his asking for a Multiple entry Visit Visa if you only know what is it :)

and yes its only for British valid for 5 years.

theres multiple entry visa.

business visit visa

tourist visa

envestor visa

working visa

and on arrival visa which is for 33 countries

and many more kind of visas but he was asking for multiple entry visa you dumb qatarisun

Figure 1: Comment with an associated Rating

3.5 Subtask B: Contrastive Runs

We only handed in contrastive runs for subtask B. The difference to our original hand-in is solely the classifier deciding whether a comment is factual or non-factual. In our first contrastive run, we used the BERT model pre-trained on the concatenation of the SemEval2016 task 3 subtask A and C data (the same we use to filter evidence). For our second run, we used a ranking model to get a similarity score between a question and a comment based on the ratings. We minimized the following

$$\text{loss} = \sum_i max(0, \delta - cos(q_i, comment_{5i}) + cos(q_i, comment_{0i}))$$

where i is a data point from the web-crawled corpus, $comment_{5i}$ is the vector obtained by the model for a comment with rating 5, $comment_{0i}$ is the obtained vector by the model for a comment without a rating, q_i is the model obtained vector for the corresponding question, δ(=0.1) is the allowed margin between a positive similarity and a negative similarity which is chosen as a hyperparameter. All vectors are obtained by max pooling the hidden states of an encoder BI-LSTM on the input text (question/comment). Our assumption is that the comment with rating 5 will be a factual answer in most of the cases (noisily labelled). Furthermore, we fine-tuned this model for the answer classification task on the training dataset for the labels 'non-factual' and 'true/false'. In table 4, we report the results for our different runs on the test set.

We also submitted an all non-factual baseline on the test set and it scored 83% accuracy. We think this biased test set hence does not reflect the model's ability to fact check comments. We reckon that in further work on this dataset, one should therefore not focus on accuracy but on a different metric.

run	**acc** (%)	F1	AvgRec	MAP
main	**0.72**	0.4	0.44	0.27
1. Contrastive run	**0.81**	0.48	0.53	0.21
2. Contrastive run	**0.48**	0.21	0.31	0.29
non-factual baseline	**0.83**	0.28	0.33	0.29

Table 4: Results of different runs for subtask B on test-set

4 Conclusion

We described our hand-in for the semeval2019 task 8. For subtask A, we fine-tuned a BERT checkpoint pretrained on a cleaned qatar living forum dump. For subtask B, we decided to use two classifiers. One classifier decides whether a comment is factual or non-factual. If it is factual, a second classifier makes a prediction about the comment's veracity. Given the small size of the training dataset, we crawled qatarliving.com to generate a medium sized, weakly supervised training corpus based on ratings in the forum. To train our model, we searched for intra-forum evidence for every comment and fine-tuned a BERT classifier for each question-comment-evidence triplet. Since the retrieved evidence is independent of each other, we did not concatenate all the evidence for a question but aggregated results for each triplet using the logsumexp function. We decided to use this function for aggregation because it allows the model to send back dense gradients and learn from all the comment-evidence pairs and not only the evidence with the highest score.

5 Acknowledgements

This work was partially supported by the German Federal Ministry of Education and Research (BMBF) through the project DEEPLEE (01IW17001).

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, Mitra Mohtarami, Georgi Karadzhov, and James R. Glass. 2018. Fact checking in community forums. In *AAAI*, pages 5309–5316. AAAI Press.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. Semeval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 525–545. Association for Computational Linguistics.

Martin F. Porter. 2001. Snowball: A language for stemming algorithms. Published online. Accessed 11.03.2008, 15.00h.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2018. The fact extraction and verification (fever) shared task. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 1–9. Association for Computational Linguistics.

Patrick Verga, Emma Strubell, and Andrew McCallum. 2018. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 872–884. Association for Computational Linguistics.

DUTH at SemEval-2019 Task 8: Part-Of-Speech Features for Question Classification

Anastasios Bairaktaris **Symeon Symeonidis** **Avi Arampatzis**

Database and Information Retrieval research unit,
Department of Electrical and Computer Engineering,
Democritus University of Thrace, Xanthi 67100, Greece

{anasbair1,ssymeoni,avi}@ee.duth.gr

Abstract

This report describes the methods employed by the Democritus University of Thrace (DUTH) team for participating in SemEval-2019 Task 8: Fact Checking in Community Question Answering Forums. Our team dealt only with Subtask A: Question Classification. Our approach was based on shallow natural language processing (NLP) preprocessing techniques to reduce noise in data, feature selection methods, and supervised machine learning algorithms such as NearestCentroid, Perceptron, and LinearSVC. To determine the essential features, we were aided by exploratory data analysis and visualizations. In order to improve classification accuracy, we developed a customized list of stopwords, retaining some opinion- and fact-denoting common function words which would have been removed by standard stoplisting. Furthermore, we examined the usefulness of part-of-speech (POS) categories for the task; by trying to remove nouns and adjectives, we found some evidence that verbs are a valuable POS category for the opinion-oriented question class.

1 Introduction

The significance of Community Question Answering (CQA) forums has risen in the past years. Such forums represent a modern need for information that comes with the abundance of online sources and the needs of millions of people for answers. Popular forums like StackOverflow, Yahoo! Answers, and Answers.com provide platforms for general or specific questions in a wide range of topics by users' and also a community-based model for user interaction.

The large numbers of questions and answers located in these forums generate many opportunities for information retrieval and data mining applications, such as query-intent detection, opinion mining, fake news classification, etc. (Tsur et al., 2016; Jo et al., 2018; Sethi, 2017). More advanced applications do not only aim at analyzing opinions but—by categorizing the feelings of the Q&As—they may be able to detect inappropriate content such as hate speech and act accordingly (Karadzhov et al., 2017; Baly et al., 2018).

The SemEval Task 8, Fact Checking in Community Forums, aims to determine whether the answers that are provided for a question in a forum are true or false. While answers to fact-oriented questions can be deemed true or false, opinion-oriented and socializing questions evoke answers for which a true/false categorization does not make much sense. As a result, determining the question type is a necessary first step. Consequently, the subtask A of SemEval Task 8 has the goal of classifying questions in three categories: opinion, factual, or socializing.

The rest of this report is structured as follows. Section 2 reviews some previous studies for CQA classification. Section 3 describes our system, while Section 4 presents experiments and results. Conclusions are summarized in Section 5.

2 Related Work

In recent years, plenty of research work examined the problem of classifying texts of CQA forums. Some related work which we found useful or inspiring are mentioned below.

Mihaylova et al. (2018) proposed a novel approach based on multi-faceted modeling of facts, which integrates knowledge from several complementary sources, such as the answer content (what is said and how), the author profile (who says it), the remainder of the community forum (where it is said), and external authoritative sources of information (external support).

Another study which provided us with helpful information about the importance of feature selection on the development of a question classifier was by Huang et al. (2008). They demonstrated the importance of using the wh-word (what, which, when) in question classification. Such words are commonly disregarded and used in stopwords lists. Our approach is also trying to use features such as imperative verbs that indicate an opinion.

The SemEval-2015 Task 3, Answer Selection in Community Question Answering, targeted to classify comments in a thread as relevant, potentially useful, or bad, concerning the thread question (Nakov et al., 2015). This task encouraged solutions for the question classification problem that involved semantic or complex linguistic information.

Finally, (Mihaylova et al., 2016; Baldwin et al., 2016; Franco-Salvador et al., 2016) participated in subtasks A, B, and C at SemEval-2016 Task 3 that involved tasks for Question-Comment Similarity, Question-Question Similarity, and Question-External Comment Similarity. They proposed classification models and provided results that highlighted the importance of lexical and semantic features.

The aforementioned studies help to identify 'gaps' in this research topic and ways to attempt new and different approaches for question classification.

3 System Description

In this section, we give the details of our question classification model, applied pre-processing techniques, as well as some statistics and visualizations for the dataset of the task.

3.1 Dataset

The organizers provided the dataset in an XML format. The given training set consisted of 1,118 questions for Subtask A that were selected from the Qatar Living forum.

We used Python's Element Tree library to parse and isolate specific content from the XML. The interesting tags to select were RelQBody (the question) and RELQ_FACT_LABEL (labeled question by organizers).

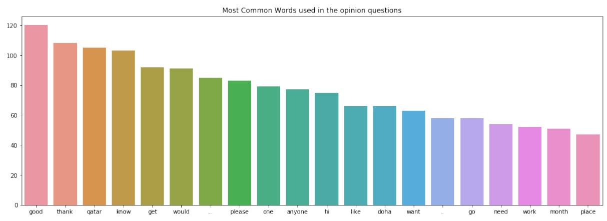
Before pre-processing, an exploratory data analysis gives us the opportunity to better understand the dataset. Because we will develop a multipurpose model that classifies not only the opinion but fact and socializing questions, it is helpful to understand in depth the character of the questions.

A way to understand the contents of the forum is to examine Table 1 where almost 50% of the questions are opinion oriented. Also, Figure 1 presents the most common words in opinion questions.

Label	**Number of Questions**
Opinion	563
Factual	311
Socializing	244

Table 1: Question types in the dataset

Figure 1: Most common words in opinion questions



3.2 Pre-processing

To reduce the noise of the text, based on the results of Symeonidis et al. (2018), we applied the following pre-processing:

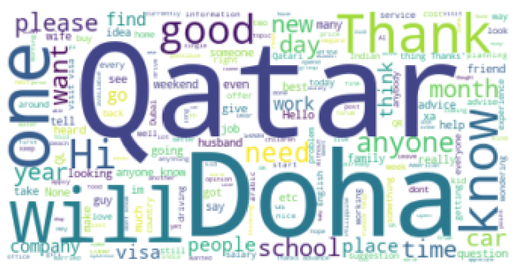
- Remove Numbers
- Remove Punctuation
- Remove Symbols
- Lowercase
- Replace all URL addresses, normalizing them to 'URL'

Figure 2 shows the most frequent words on the dataset as a wordcloud.

The final steps of pre-processing are tokenization and stemming. A basic process in NLP is to identify tokens or those basic units which need not be decomposed in subsequent processing.

The entity word is one kind of token for NLP (Webster and Kit, 1992). Stemming is a process of reducing words to their stems or roots to reduce the vocabulary size and manage the case of data sparseness (Lin and He, 2009). For example, conjugated verbs such as 'goes', 'going', and 'gone' are stemmed to the term 'go'.

Figure 2: Wordcloud of frequent words



We used Python's SpaCy[1] library to tokenize the text and convert it to lemmas. This function also removes symbols (or punctuation) such as '[','...','-'.

Stopwords are frequent words that appear in the text, but they can have an impact on retrieval efficacy. The removal of stopwords also modifies the document length and subsequently affects the weighting process and efficiency during processing of the collection (Kwok, 1998).

For our task, we found out that the most commonly-used stopword lists contain words that can be helpful. For example, the word 'believe' is included in most stopword lists. While it is a ubiquitous word, it may also indicate an opinion; therefore, it can be useful for our purpose. In order to tackle this problem, we made a custom stopword list that only removes pronouns such as 'i','he','she', etc. NLTK's[2] list of English stopwords used as guideline and contained 127 words.

Although there is an abundance of stopwords lists that contain even more words we used a small one on purpose. We wanted to eliminate words from our dataset that would not bear any significance in our task. The next step, based on the vocabulary of the dataset, was to manually find words that could help us identify whether the question is opinion oriented, factual, or socializing. We excluded, from the NLTK's stopword list, words such as 'believe', 'think', 'mean', 'consider', and others. Our final revised stopword list consists of 50 words.

4 Experiments

This section summarizes our experiments in the context of SemEval 2019 Task 8 Subtask A. Beyond our officially submitted runs, we present some additional experiments that although they did not perform very well, there seems to be a promising room for improvement in the future.

[1]https://spacy.io

[2]https://www.nltk.org/

4.1 Machine Learning Methods

For the training of our classifiers, we used Python's Scikit-Learn library (Pedregosa et al., 2011). We split the dataset into 749 training questions and 369 testing questions, i.e. a typical 2/3–1/3 split (ratio 2:1). After the split, the questions in the training set were shuffled for training. With the class `sklearn.pipeline`, we performed a sequence of different transformations and parameters.

Vectorizer: We compared three common vectorizers such as CountVectorizer, HashingVectorizer, and TfidfVectorizer. Finally, our selection was the TfidfVectorizer since it yielded the best results when it comes to accuracy. The TfidfVectorizer converts a collection of raw documents to a matrix of tf-idf weighted features.

Classifiers: We experimented with various classifiers, and decided to use the following three since they yielded the best accuracy results.

- **NearestCentroid**: Each class is represented by its centroid, with test samples classified to the class with the nearest centroid.
- **Perceptron**: It is a simple and efficient algorithm to fit linear models, and suitable for very large numbers of features.
- **LinearSVC**: An SVM algorithm which tries to find a set of hyperplanes that separate space into areas representing the classes. The hyperplanes are chosen in a way to maximize the distance from the nearest data point of each class.

4.2 Results

The official run that we submitted for the competition proved to be the most successful. In the following tables, we present the produced test results by using the three different classifiers, the TfidfVectorizer, and the custom stopword list. The results are shown in Tables 2, 3, and 4. We can observe that the most accurate classifier overall is the NearestCentroid.

We experimented further based on the hypothesis that opinion classification can be more effective by using only verbs. Until recently, most

LinearSVC	accuracy	recall	f1-score
Factual	0.60	0.40	0.48
Opinion	0.62	0.82	0.70
Socializing	0.75	0.53	0.62
Total	0.64	0.64	0.62

Table 2: Test results with LinearSVC

NearestCentroid	accuracy	recall	f1-score
Factual	0.63	0.49	0.55
Opinion	0.71	0.76	0.73
Socializing	0.70	0.77	0.73
Total	0.68	0.69	0.68

Table 3: Test results with NearestCentroid

classification techniques have considered adjectives, adverbs, and nouns as features. The usefulness of part-of-speech categories in text classification was investigated as early as in (Arampatzis et al., 2000), where it was found that a traditional keyword-based indexing set can be reduced to retain only its nouns and adjectives without hurting effectiveness, even slightly improving it. Nevertheless, the aforementioned work was on topic classification; later, Karamibekr and Ghorbani (2012) showed that verbs are vital in classifying opinion terms, particularly in social domains.

We conducted two experiments by removing either nouns or adjectives from our dataset to help our classifiers adjust mostly on verbs. We can observe, in Tables 5 and 6, that classifiers achieved a better accuracy score when it comes to opinion as opposed to fact and socializing questions. Nevertheless, by removing either nouns or adjectives, there is an overall drop in effectiveness in all classes. Thus, there is evidence that verbs are a useful part-of-speech category for opinion classification, but they are not sufficient by themselves.

Our official submission to the competition ranked our team to the 16th place from 22 teams.

Perceptron	accuracy	recall	f1-score
Factual	0.54	0.36	0.43
Opinion	0.62	0.79	0.70
Socializing	0.64	0.49	0.56
Total	0.60	0.61	0.59

Table 4: Test results with Perceptron

NearestCentroid	accuracy	recall	f1-score
Factual	0.52	0.42	0.46
Opinion	0.68	0.66	0.67
Socializing	0.56	0.72	0.63
Total	0.61	0.61	0.60

Table 5: Test results without Nouns

NearestCentroid	accuracy	recall	f1-score
Factual	0.51	0.40	0.45
Opinion	0.66	0.65	0.66
Socializing	0.53	0.69	0.60
Total	0.59	0.59	0.59

Table 6: Test results without Adjectives

The results of our model are shown in Table 7.

Accuracy	F1	AverageRecall
0.71	0.56	0.60

Table 7: Official Results - Use of NearestCentroid

5 Conclusions

We presented a supervised learning model for classifying questions from online Q&A forums in three categories: factual, opinion, and socializing. We used standard pre-processing techniques, and made a custom stopword list to tackle the specific task at hand. Using standard classification methods, we achieved satisfactory and promising results. We also tried to use verb-oriented feature sets for classification which although they provided mixed results it seems that they can be improved.

References

Avi Arampatzis, Th.P. van der Weide, C.H.A. Koster, and P. van Bommel. 2000. An evaluation of linguistically-motivated indexing schemes. In *Proceedings of the 22nd BCS-IRSG Colloquium on IR Research*, pages 34–45.

Timothy Baldwin, Huizhi Liang, Bahar Salehi, Doris Hoogeveen, Yitong Li, and Long Duong. 2016. Unimelb at semeval-2016 task 3: Identifying similar questions by combining a CNN with string similarity measures. In (Bethard et al., 2016), pages 851–856.

Ramy Baly, Mitra Mohtarami, James R. Glass, Lluís Màrquez, Alessandro Moschitti, and Preslav Nakov.

2018. Integrating stance detection and fact checking in a unified corpus. *CoRR*, abs/1804.08012.

Steven Bethard, Daniel M. Cer, Marine Carpuat, David Jurgens, Preslav Nakov, and Torsten Zesch, editors. 2016. *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*. The Association for Computer Linguistics.

Marc Franco-Salvador, Sudipta Kar, Thamar Solorio, and Paolo Rosso. 2016. UH-PRHLT at semeval-2016 task 3: Combining lexical and semantic-based features for community question answering. In (Bethard et al., 2016), pages 814–821.

Zhiheng Huang, Marcus Thint, and Zengchang Qin. 2008. Question classification using head words and their hypernyms. In *2008 Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, Proceedings of the Conference, 25-27 October 2008, Honolulu, Hawaii, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 927–936. ACL.

Saehan Jo, Immanuel Trummer, Weicheng Yu, Daniel Liu, and Niyati Mehta. 2018. The factchecker: Verifying text summaries of relational data sets. *CoRR*, abs/1804.07686.

Georgi Karadzhov, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, and Ivan Koychev. 2017. Fully automated fact checking using external sources. *CoRR*, abs/1710.00341.

Mostafa Karamibekr and Ali A. Ghorbani. 2012. Verb oriented sentiment classification. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence, WI 2012, Macau, China, December 4-7, 2012*, pages 327–331. IEEE Computer Society.

K. L. Kwok. 1998. Book review: Information storage and retrieval by r. r. korfhage. *Inf. Process. Manage.*, 34(4):490–492.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 375–384. ACM.

Tsvetomila Mihaylova, Pepa Gencheva, Martin Boyanov, Ivana Yovcheva, Todor Mihaylov, Momchil Hardalov, Yasen Kiprov, Daniel Balchev, Ivan Koychev, Preslav Nakov, Ivelina Nikolova, and Galia Angelova. 2016. Super team at semeval-2016 task 3: Building a feature-rich system for community question answering. In (Bethard et al., 2016), pages 836–843.

Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, Mitra Mohtarami, Georgi Karadzhov, and James R. Glass. 2018. Fact checking in community forums. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5309–5316. AAAI Press.

Preslav Nakov, Lluís Màrquez, Walid Magdy, Alessandro Moschitti, Jim Glass, and Bilal Randeree. 2015. Semeval-2015 task 3: Answer selection in community question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015*, pages 269–281. The Association for Computer Linguistics.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830.

Ricky J. Sethi. 2017. Crowdsourcing the verification of fake news and alternative facts. In *Proceedings of the 28th ACM Conference on Hypertext and Social Media, HT 2017, Prague, Czech Republic, July 4-7, 2017*, pages 315–316. ACM.

Symeon Symeonidis, Dimitrios Effrosynidis, and Avi Arampatzis. 2018. A comparative evaluation of preprocessing techniques and their interactions for twitter sentiment analysis. *Expert Syst. Appl.*, 110:298–310.

Gilad Tsur, Yuval Pinter, Idan Szpektor, and David Carmel. 2016. Identifying web queries with question intent. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 783–793. ACM.

Jonathan J. Webster and Chunyu Kit. 1992. Tokenization as the initial phase in NLP. In *14th International Conference on Computational Linguistics, COLING 1992, Nantes, France, August 23-28, 1992*, pages 1106–1110.

Fermi at SemEval-2019 Task 8: An elementary but effective approach to Question Discernment in Community QA Forums

Bakhtiyar Syed[1], Vijayasaradhi Indurthi[1,3], Manish Shrivastava[1], Manish Gupta[1,2], Vasudeva Varma[1]

[1] IIIT Hyderabad, [2] Microsoft, [3] Teradata

[1]{syed.b, vijaya.saradhi}@research.iiit.ac.in

[1]{m.shrivastava, manish.gupta, vv}@iiit.ac.in

[2]gmanish@microsoft.com

[3]vijayasaradhi.indurthi@teradata.com

Abstract

Online Community Question Answering Forums (cQA) have gained massive popularity within recent years. The rise in users for such forums have led to the increase in the need for automated evaluation for question comprehension and fact evaluation of the answers provided by various participants in the forum. Our team, **Fermi**, participated in sub-task A of Task 8 at SemEval 2019 - which tackles the first problem in the pipeline of factual evaluation in cQA forums, i.e., deciding whether a posed question asks for a factual information, an opinion/advice or is just socializing. This information is highly useful in segregating factual questions from non-factual ones which highly helps in organizing the questions into useful categories and trims down the problem space for the next task in the pipeline for fact evaluation among the available answers. Our system uses the embeddings obtained from Universal Sentence Encoder combined with XGBoost for the classification sub-task A. We also evaluate other combinations of embeddings and off-the-shelf machine learning algorithms to demonstrate the efficacy of the various representations and their combinations. Our results across the evaluation *test* set gave an accuracy of 84% and received the **first** position in the final standings judged by the organizers.

1 Introduction

The massive rise in popularity of Community Question Answering (cQA) forums like StackOverflow, Quora, Yahoo! Answers and Google Groups have led to an effective means of information dissemination for topic-centered communities to share and engage in knowledge consumption needs. After a considerable time, information becoming obsolete is a major problem which results in change of many of the facts that were previously true. Another problem is that most of the forums lack exhaustive moderation and control – which results in high-latency quality checks and eventually results in the sharing of non-factual information. Various factors are responsible for this – primarily being ignorance or misunderstanding and sometimes, maliciousness of the responder to the questions (Mihaylova et al., 2018).

In the pipeline of detection of whether the given responses to a question are indeed factual, the necessary first step is to discern what category the question asked in the cQA forum falls into. As an example, *"What is Domino's customer service number?"* is a factual question as it asks for a fact rather than an opinion or discourse. In contrast, consider the question *"Can someone recommend a good pediatrician in Mumbai?"* asks for an opinion rather than a particular factual information as opinions on the matter of a good pediatrician may be subjective and depend on various other factors the conclusion of which is not universally true.

We tackle the problem proposed by organizers (Mihaylova et al., 2019) in sub-task A as a multi-class classification problem, i.e., categorizing questions in cQA forums into one of the following three categories:

1. Factual: The question is asking for factual information, which can be answered by checking various information sources, and it is not ambiguous. *(e.g., "What is the currency used in Taiwan?")*

2. Opinion: The question asks for an opinion or an advice, not for a fact. *(e.g., "Can somebody recommend good restaurants around the SF Bay Area?")*

3. Socializing: Not a real question, but intended for socializing or for chatting. This can also mean expressing an opinion or sharing some information, without really asking anything of general interest. *(e.g., "What was your first bike?")*

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 1160–1164
Minneapolis, Minnesota, USA, June 6–7, 2019. ©2019 Association for Computational Linguistics

Our submission involves the use of pre-trained models for generating sentence embeddings from existing trained models and then employing the use of off-the-shelf machine learning algorithms for the multi-class prediction problem. The approach is described in Section 3 where we describe our methodology in detail.

2 Related Work

For classification tasks like question similarity across community QA forums, machine learning classification algorithms like Support Vector Machines (SVMs) have been used (Šaina et al., 2017; Nandi et al., 2017; Xie et al., 2017; Mihaylova et al., 2016; Wang and Poupart, 2016; Balchev et al., 2016). Recently, advances in deep neural network architectures have also led to the use of Convolutional Neural Networks (CNNs) (Šaina et al., 2017; Mohtarami et al., 2016) which perform reasonably well for selection of the correct answer amongst cQA formus. Algorithms and methods for answer selection also include works by (Zhang et al., 2017) which use a Long-Short Term Memory (LSTM) model for answer selection. Similarly, LSTMs for answer selection are also used by (Feng et al., 2017; Mohtarami et al., 2016). Other works in the space include use of Random Forests (Wang and Poupart, 2016); topic models to match the questions at both the term level and topic level (Zhang et al., 2014). There have also been works on translation based retrieval models (Jeon et al., 2005; Zhou et al., 2011); XgBoost (Feng et al., 2017) and Feedforward Neural Networks (NN) (Wang and Poupart, 2016).

All of the above related works on cQA used the features such as Bag of Words (BoW) (Franco-Salvador et al., 2016); Bag of vectors (BoV) (Mohtarami et al., 2016); Lexical features (for example, Cosine Similarity, Word Overlap, Noun Overlap, N-gram Overlap, Longest Common Substring/Subsequence, Keyword and Named Entity features etc.) (Franco-Salvador et al., 2016; Mohtarami et al., 2016; Nandi et al., 2017); Semantic features (for example, Distributed representations of text, Knowledge Graphs, Distributed word alignments, Word Cluster Similarity, etc.) (Franco-Salvador et al., 2016); Word Embedding Features (like Word2vec, GloVe etc.) (Wang and Poupart, 2016; Mohtarami et al., 2016; Nandi et al., 2017); and Metadata-based features (Mohtarami et al., 2016; Mihaylova et al., 2016; Xie et al., 2017).

In this work, we seek to evaluate pre-trained sentence embeddings and how they perform across comprehension of questions in the community QA tasks. We now describe the methodology and data in the following section.

3 Methodology and Data

The data supplied by organizers is used for the task at hand. Specifically, for sub-task A, the *subject* and *body* for each question is provided by the task organizers. The data consists of 1118 training instances along with 239 and 935 question instances in the development and testing sets respectively.

3.1 Word Embeddings

Word embeddings have been widely used in modern Natural Language Processing applications as they provide vector representation of words. They capture the semantic properties of words and the linguistic relationship between them. These word embeddings have improved the performance of many downstream tasks across many domains like text classification, machine comprehension etc. (Camacho-Collados and Pilehvar, 2018). Multiple ways of generating word embeddings exist, such as Neural Probabilistic Language Model (Bengio et al., 2003), Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and more recently ELMo (Peters et al., 2018).

These word embeddings rely on the distributional linguistic hypothesis. They differ in the way they capture the meaning of the words or the way they are trained. Each word embedding captures a different set of semantic attributes which may or may not be captured by other word embeddings. In general, it is difficult to predict the relative performance of these word embeddings on downstream tasks. The choice of which word embeddings should be used for a given downstream task depends on experimentation and evaluation.

3.2 Sentence Embeddings

While word embeddings can produce representations for words which can capture the linguistic properties and the semantics of the words, the idea of representing sentences as vectors is an important and open research problem (Conneau et al., 2017).

Finding a universal representation of a sentence which works with a variety of downstream tasks is the major goal of many sentence embedding techniques. A common approach of obtaining a sentence representation using word embeddings is by the simple and naïve way of using the simple arithmetic mean of all the embeddings of the words present in the sentence. Smooth inverse frequency, which uses weighted averages and modifies it using Singular Value Decomposition (SVD), has been a strong contender as a baseline over traditional averaging technique (Arora et al., 2016). Other sentence embedding techniques include p-means (Rücklé et al., 2018), InferSent (Conneau et al., 2017), SkipThought (Kiros et al., 2015), Universal Encoder (Cer et al., 2018).

We formulate sub-task A of Task 8 in SemEval 2019 as a text multi-classification task. In this paper, we evaluate various pre-trained sentence embeddings for identifying each of the categories of factual, socializing and opinion among the questions in community QA forums. We train multiple models using different machine learning algorithms to evaluate the efficacy of each of the pre-trained sentence embeddings for the sub-task. In the following, we discuss various popular sentence embedding methods in brief.

- InferSent (Conneau et al., 2017) is a set of embeddings proposed by Facebook. InferSent embeddings have been trained using the popular language inference corpus. Given two sentences the model is trained to infer whether they are a contradiction, a neutral pairing, or an entailment. The output is an embedding of 4096 dimensions.

- Concatenated Power Mean Word Embedding (Rücklé et al., 2018) generalizes the concept of average word embeddings to power mean word embeddings. The concatenation of different types of power mean word embeddings considerably closes the gap to state-of-the-art methods mono-lingually and substantially outperforms many complex techniques cross-lingually.

- Lexical Vectors (Salle and Villavicencio, 2018) is another word embedding similar to fastText with slightly modified objective. FastText (Bojanowski et al., 2016) is another word embedding model which incorporates character n-grams into the skipgram model of Word2Vec and considers the sub-word information.

- The Universal Sentence Encoder (Cer et al., 2018) encodes text into high dimensional vectors. The model is trained and optimized for greater-than-word length text, such as sentences, phrases or short paragraphs. It is trained on a variety of data sources and a variety of tasks with the aim of dynamically accommodating a wide variety of natural language understanding tasks. The input is variable length English text and the output is a 512 dimensional vector.

- Deep Contextualized Word Representations (ELMo) (Peters et al., 2018) use language models to get the embeddings for individual words. The entire sentence or paragraph is taken into consideration while calculating these embedding representations. ELMo uses a pre-trained bi-directional LSTM language model. For the input supplied, the ELMo architecture extracts the hidden state of each layer. A weighted sum is computed of the hidden states to obtain an embedding for each sentence.

Using each of the sentence embeddings we have mentioned above, we seek to evaluate how each of them performs when the vector representations of the body of questions in the cQA forums are supplied for classification with various off-the-shelf machine learning algorithms. For each of the evaluation tasks, we perform experiments using each of the sentence embeddings mentioned above and show our classification performance on the *dev* set given by the task organizers.

Model	**F-1**	**Acc**
Universal Encoder + XGB	**0.72**	**0.84**

Table 1: Results showing Macro-F1 score and accuracy for Sub-task A, using Universal Encoder Sentence embeddings and training the model with XGBoost.

4 Results

The official ranking metric is Accuracy. We have included the F-1 score here as well for comparison. Table 1 provides the results on the system runs for the evaluation phase as judged by the organizers on the CodaLab platform. Our system ranked *first* among the participants in the evaluation phase. We observe that Universal Sentence

Model	**RF**		**SVM-RBF**		**XGBoost**	
	Acc.	F-1	Acc.	F-1	Acc.	F-1
Universal Sentence Encoder	68.66	72.32	67.38	68.25	73.73	73.56
InferSent	53.91	50.89	61.56	63.45	60.82	59.32
Concat-p mean	56.22	49.01	65.64	69.54	60.36	60.01
Lexical Vectors	62.80	62.11	72.42	71.55	71.30	68.30

Table 2: *Dev* Set Accuracy and Macro-F-1 scores (in percentage) for **Sub-Task A** of Task 8

Encoder representations with the XGBoost classifier gives the best results on the test set.

As a way to elicit different performances for our experiments, we also provide our results from the system runs on the development set provided by the organizers. These results are shown in Table 2.

5 Conclusions and Future Work

We see from the results that our system is able to discern the type of questions asked in community QA forums with high performance metrics. This shows that using pre-trained embeddings with a simple machine learning classification algorithm often helps in greater understanding of the text at hand – in this case, the questions in community question-answering forums.

In future work, we also seek to evaluate different transfer learning approaches which utilize pre-trained language models (LMs) across different base language corpora and see how varying these base corpora for pre-training the language model results in the performance change while finetuning for question comprehension in cQA forums.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings.

Daniel Balchev, Yasen Kiprov, Ivan Koychev, and Preslav Nakov. 2016. PMI-cool at SemEval-2016 Task 3: Experiments with PMI and Goodness Polarity Lexicons for Community Question Answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 844–850, San Diego, California. Association for Computational Linguistics.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Jose Camacho-Collados and Mohammad Taher Pilehvar. 2018. From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research*, 63:743–788.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Wenzheng Feng, Yu Wu, Wei Wu, Zhoujun Li, and Ming Zhou. 2017. Beihang-MSRA at SemEval-2017 Task 3: A Ranking System with Neural Matching Features for Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 280–286, Vancouver, Canada. Association for Computational Linguistics.

Marc Franco-Salvador, Sudipta Kar, Thamar Solorio, and Paolo Rosso. 2016. UH-PRHLT at SemEval-2016 Task 3: Combining Lexical and Semantic-based Features for Community Question Answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 814–821, San Diego, California. Association for Computational Linguistics.

Jiwoon Jeon, W Bruce Croft, and Joon Ho Lee. 2005. Finding Similar Questions in Large Question and Answer Archives. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 84–90. ACM.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Tsvetomila Mihaylova, Pepa Gencheva, Martin Boyanov, Ivana Yovcheva, Todor Mihaylov, Momchil Hardalov, Yasen Kiprov, Daniel Balchev, Ivan Koychev, Preslav Nakov, Ivelina Nikolova, and Galia

Angelova. 2016. SUper Team at SemEval-2016 Task 3: Building a Feature-Rich System for Community Question Answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 836–843, San Diego, California. Association for Computational Linguistics.

Tsvetomila Mihaylova, Georgi Karadzhov, Atanasova Pepa, Ramy Baly, Mitra Mohtarami, and Preslav Nakov. 2019. SemEval-2019 task 8: Fact checking in community question answering forums. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '19, Minneapolis, MN, USA.

Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez i Villodre, Alberto Barrón-Cedeño, Mitra Mohtarami, Georgi Karadzhov, and James R. Glass. 2018. Fact checking in community forums. In *AAAI*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mitra Mohtarami, Yonatan Belinkov, Wei-Ning Hsu, Yu Zhang, Tao Lei, Kfir Bar, Scott Cyphers, and Jim Glass. 2016. SLS at SemEval-2016 Task 3: Neural-based Approaches for Ranking in Community Question Answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 828–835, San Diego, California. Association for Computational Linguistics.

Titas Nandi, Chris Biemann, Seid Muhie Yimam, Deepak Gupta, Sarah Kohail, Asif Ekbal, and Pushpak Bhattacharyya. 2017. IIT-UHH at SemEval-2017 Task 3: Exploring Multiple Features for Community Question Answering and Implicit Dialogue Identification. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 90–97, Vancouver, Canada. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Andreas Rücklé, Steffen Eger, Maxime Peyrard, and Iryna Gurevych. 2018. Concatenated p-mean word embeddings as universal cross-lingual sentence representations. *arXiv preprint arXiv:1803.01400*.

Alexandre Salle and Aline Villavicencio. 2018. Incorporating subword information into matrix factorization word embeddings. *arXiv preprint arXiv:1805.03710*.

Filip Šaina, Toni Kukurin, Lukrecija Puljić, Mladen Karan, and Jan Šnajder. 2017. TakeLab-QA at SemEval-2017 Task 3: Classification Experiments for Answer Retrieval in Community QA. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 339–343, Vancouver, Canada. Association for Computational Linguistics.

Hujie Wang and Pascal Poupart. 2016. Overfitting at SemEval-2016 Task 3: Detecting Semantically Similar Questions in Community Question Answering Forums with Word Embeddings. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 861–865, San Diego, California. Association for Computational Linguistics.

Yufei Xie, Maoquan Wang, Jing Ma, Jian Jiang, and Zhao Lu. 2017. EICA Team at SemEval-2017 Task 3: Semantic and Metadata-based Features for Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 292–298, Vancouver, Canada. Association for Computational Linguistics.

Kai Zhang, Wei Wu, Haocheng Wu, Zhoujun Li, and Ming Zhou. 2014. Question Retrieval with High Quality Answers in Community Question Answering. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 371–380. ACM.

Sheng Zhang, Jiajun Cheng, Hui Wang, Xin Zhang, Pei Li, and Zhaoyun Ding. 2017. FuRongWang at SemEval-2017 Task 3: Deep Neural Networks for Selecting Relevant Answers in Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 320–325, Vancouver, Canada. Association for Computational Linguistics.

Guangyou Zhou, Li Cai, Jun Zhao, and Kang Liu. 2011. Phrase-Based Translation Model for Question Retrieval in Community Question Answer Archives. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 653–662. Association for Computational Linguistics.

SolomonLab at SemEval-2019 Task 8: Question Factuality and Answer Veracity Prediction in Community Forums

Sudeep Kumar Sahoo* **Ankita Gupta*** **Divya Prakash**
Rohit R.R **Vertika Srivastava** **Yeon Hyang Kim**

Samsung R&D Institute India, Bangalore
{gupta.ankita, sudeep.sahoo, p.divya, rohit.r.r, v.srivastava, purine.kim}@samsung.com

Abstract

We describe our system for SemEval-2019, Task 8 on "Fact-Checking in Community Question Answering Forums (cQA)". cQA forums are very prevalent nowadays, as they provide an effective means for communities to share knowledge. Unfortunately, this shared information is not always factual and fact-verified. In this task, we aim to identify factual questions posted on cQA and verify the veracity of answers to these questions. Our approach relies on data augmentation and aggregates cues from several dimensions such as semantics, linguistics, syntax, writing style and evidence obtained from trusted external sources. In subtask A, our submission is ranked 3rd, with an accuracy of 83.14%. Our current best solution stands 1st on the leaderboard with 88% accuracy. In subtask B, our present solution is ranked 2nd, with 58.33% MAP score.

1 Introduction

With the rising popularity of online community question answering (cQA) systems such as Quora, StackOverflow, and Qatar Living forum (QLF), the amount of information shared over these platforms is also increasing rapidly with time. These forums provide effective information sharing mechanism to their users who can seek answers to their queries as well as post answers to the questions. However, the information shared on such platforms may not always be factual and correct. The responders may misunderstand the question being asked or merely ignore certain specific details. At times, the information shared may even be false or ambiguous in the desired context. This is aggravated by lack of moderation and systematic control on cQA forums. The Semeval-2019 Task 8[1] on "Fact Checking in Community Question Answering Forums" aims to solve this real-life problem.

The above task tries to explore the veracity of an answers to a question posted on QLF. While the precedent tasks such as SemEval (Nakov et al., 2015, 2016, 2017), address the issue of ranking answers according to their relevance to a question, the task-at-hand is the first one to consider the correctness of answers. This task is formulated as a two-stage problem. The first stage aims to identify the user posts asking for factual information. The answers to the identified factual questions are then fact-verified in the second stage. Both the subtasks are designed as 3-class supervised classification problems.

More specifically, the first stage or subtask A addresses the problem of determining whether the posted question asks for factual information, an opinion/advice or is just meant for socializing. For example, *"what is Ooredoo customer service number?"* asks for factual information, whereas *"What was your first car?"* is socializing and *"which is the best bank around?"* is seeking guidance/opinion. Each data sample in subtask A is a question posted by a user consisting of a subject, body and meta information (user ID, username, and the category of question, e.g., "Education," "Visa and Permits", "Welcome to Qatar" etc.).

The second stage or subtask B focuses on determining whether an answer to a factual question is true, false or does not constitute a proper answer, in which case, it is labeled as non-factual. For example, to the question *"Can I bring my pitbulls to Qatar?"*, Answer A1: *"Yes, you can bring it but be careful this kind of dog is very dangerous"* is factual-false[2], Answer A2: *"No, you cannot as they are banned"* is factual-true[2] and Answer A3: *"There goes another job opportunity for the sake*

*Equal Contribution
[1]http://alt.qcri.org/semeval2019/index.php?id=tasks

[2]can be verified at http://canvethospital.com/pet-relocation/

of two lovely animals. ” is non-factual. The data is organized as a question-answer tuple: question posted by a user and an answer (body, username and answer ID) posted by the same or another user. It has been ensured that all the questions in this task are factual questions.

Our approach to solving this task is based on extracting rich-feature representation from the input and training a classifier to make predictions. The feature representation integrates knowledge from various complementary sources, such as the question/answer content, the content of other answers in the thread, evidence from trustworthy external sources of information, and the relevance of an answer to the question. For subtask A, we rely on question content (semantic, linguistic and syntactic cues), whereas the evidence from external sources and answer relevancy to the question are essential aspects for subtask B. For both the subtasks, we also leverage a data augmentation approach which facilitates the generalization ability of learned classifier on unseen test data as well as ameliorates the class imbalance issues present in the training data.

The rest of the paper is organized as follows: Section 2 gives an overview of our system. Section 3-5 describe the details of our approach. Section 6 demonstrates the experimental results. We conclude in Section 7.

2 System Overview

Our proposed system primarily relies on following key components (i) data augmentation (DA) (ii) pre-processing of question/answer content and (iii) feature extraction from multi-faceted sources.

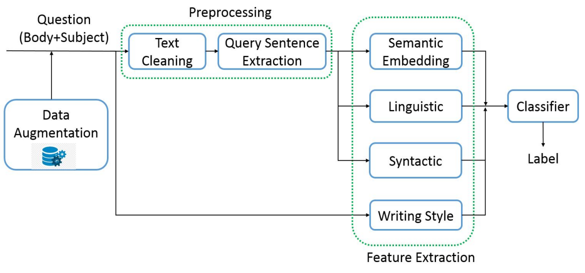


Figure 1: System Overview for Subtask A

As depicted in Figure 1, following DA and preprocessing of the question, our system for subtask A extracts semantic (what is said), linguistic (how it is said), syntactic (how it is structured) and writing style based features (how it is depicted) from the processed question. These extracted features are then combined to train a classifier for label prediction.

Subtask B also leverages DA and preprocessing as its first key steps. However, apart from features extracted for subtask A (as mentioned above), it also utilizes external evidence and forum-level features (Figure 2). The external evidence is collected from trusted sources using a search-engine. The forum-level features capture the relevancy of an answer to the question and its similarity to other answers in the same thread.

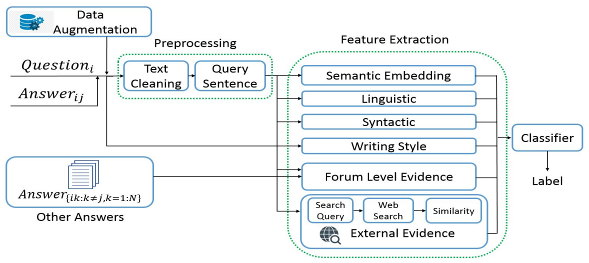


Figure 2: System Overview for Subtask B

3 Data Augmentation (DA)

Data augmentation (DA) is one of the main components of our proposed system that resulted in significant performance gains. For both the subtasks, the training data is imbalanced. This motivated us to look for ways to balance the distribution of data samples across classes and at the same time incorporate adversarial examples which are plausible in the real scenario but are not present in the training data. We next discuss the DA details for both the subtasks in the following subsections.

3.1 Subtask A

In the training data for subtask A, the number of samples from the “opinion” class (563) is observed to be twice as many samples from “factual” (311) or “socializing” class (244). In order to balance the class distribution, we sought to oversample both of the non-majority classes based on the domain knowledge.

For the “Factual” class, we leveraged the questions asked in subtask B. In subtask B, by its formulation, one is supposed to verify the veracity of answers to “factual questions.” Thus, we used the training, development and test set of subtask B to augment training data for subtask A (“factual” class instances). This way, we extracted a total of 91 distinct factual questions. For the “socializing” class, we utilized the QL-unannotated-data[3] to se-

[3]additional resource by (Mihaylova et al., 2018)

Class	Question Body
Factual	**Can someone please tell me where can i find Garlic Oil in Qatar?** i heard it is good for hairfall. dont know if its true or not but really want to try it. So help me guys!
Opinion	**Is it right to resign from your job at this time of global crisis?** the reason is i'm not doing anything in the office. I feel useless; but I'm hesistant to resign because of the condition today even that I'm on husband sponsor.
Socializing	**Is this a beginning of a mutual friendship between Christianity and Islam in Qatar?** I hope they're going to sell some Bibles in Villagio coz I can't find somebody sellin' it around here.

Table 1: Example for query-sentence selection. The highlighted text is considered as the query-sentence.

lect samples from categories ("Funnies," "Good News Everyone," "Party on my mind," "Recipes," "Press Releases") that are assured to contain only socializing content. In these categories, the users are just trying to make conversation or share anecdotes. As the number of such samples is considerably large, we sample 320 samples (using reservoir sampling (Vitter, 1985)) to balance the distribution across classes in the original training data.

3.2 Subtask B

For subtask B, we consider an adversarial setting closely related to the problem at hand. As mentioned before, each data sample in this subtask is a question-answer tuple, and the answer can be either "true", "false" or "non-factual." A related task was demonstrated in Semeval 2016 task 3 "Answer Selection in cQA" (Nakov et al., 2016) where the objective was to re-rank the answers based on their relevancy to the question. In this task, the replies such as follow-up question from other user, clarifications, and acknowledgment from the user himself were categorized as "Bad" answers. Although, in the task-at-hand, the organizers have omitted such answers, in the real-life scenario they will also be present and should be categorized as "non-factual" in our current problem setting.

Thus, to include such samples, we extract factual questions from the training data of subtask A. For each of these questions, we select "bad answers" from the data provided in the SemEval 2016 task. The chosen question-answer pair is then annotated as "non-factual" and added to the training data of subtask B.

4 Preprocessing

Before feature extraction, we pre-process the input question/answer using several key steps. We expand the contractions and terms commonly used on social media platforms such as 'i'm: 'i am,' 'i'd: i would,' 'pls: please,' 'nt: not,' 'thru: through' etc. Furthermore, we use several markers such as URLs, images, emoticons, and punctuation marks in the question/answer to extract writing style and syntactic features (described in Section 5.3). For semantic and linguistic features, we strip these markers.

4.1 Query Sentence Extraction

Based on the empirical evidence, we could infer that the body of each question posed by the user contains several sentences. However, among all these sentences, only one or two convey the query he/she really wants to ask. Also, the user may post his question in the question subject itself. Thus, we extract these "query-sentences" from the question body and subject and use them to extract linguistic, semantic features. An example of the query-sentence and original question posted by the user for each of the three classes corresponding to subtask A is depicted in Table 1.

In order to extract query-sentence, we parse each sentence in the question using Stanford CoreNLP constituency parser (Manning et al., 2014). A sentence is considered a query-sentence if its parse-tree has *SBARQ/SQ* constituent phrases. We also use some common heuristics such as, whether the sentence ends with a question-mark or starts with common "wh" words (what, why, how etc).

5 Modeling Content : Feature Extraction

We use rich feature representation to model the information conveyed in question/answer. In the subsequent subsections, we describe the details of each of these features.

5.1 Semantic Sentence Embedding

Following the pre-processing step, we compute semantic sentence embedding for query-sentence by using two approaches. The first approach utilizes universal sentence encoder (USE) (Cer et al., 2018). It has been known to perform well with minimal amounts of supervised training data for a downstream task, which is precisely our setting for both the subtasks. The second approach appropriates pre-trained word embeddings (glove) (Pennington et al., 2014), averaged over each word in a sentence to compute sentence-level embedding.

5.2 Linguistic Features

Often, forum users exhibit linguistic cues in writing questions and answers. For example, they may use subjectively loaded words such as 'awesome,' 'worst' etc. while asking for an opinion rather than factual information. While answering on the forum, they may exhibit the degree of confidence in the truthfulness of what they say by using words like "most likely", "probably", "think" etc. We therefore use linguistic markers such as hedges (Hyland, 2018), weasels(Vincze, 2013), factives (Hooper, 1974), assertives (Hooper, 1974), implicatives (Karttunen, 1971), mood[4], modality[4], subjectivity[4], sentiment[4] and polarity of subjective words (Riloff and Wiebe, 2003) based on respective lexicons to compute a feature vector. (For details, refer to (Mihaylova et al., 2018))

5.3 Writing Style Features

We extract writing style features from the question/answer which capture the format of a user-post. A socializing question is more likely to be written informally as compared to factual/opinion query. A non-factual answer which is not much informative may also carry distinctive cues. To capture these aspects, we count the number of punctuations, emoticons, NON-ASCII characters and check the presence of URL, image, ALL CAPS, consecutive character repetition ($\geq$ 3 times). Table 2 depicts how the number of samples exhibiting a particular writing style feature vary across the three classes in subtask A. A similar trend is present for factual (true/false) versus non-factual answers in subtask B.

Feature	Opinion	Factual	Socializing
All Caps	8	1	4
URL	2	0	173
Image	0	0	3
Repetition	9	3	130

Table 2: Writing Style Based Feature Distribution across Classes for Subtask A

5.4 Syntactic Features

We also examine syntactic features such as part-of-speech (POS) and category of question encoded as bag-of-words features. Further, we consider the expected answer type for a question (QType) and named-entity-type (NET) in an answer.

QType suggests the kind of information the question is seeking such as "description", "entity", "human", "location", "number", "yes/no" and "others" (extracted using work in (Madabushi and Lee, 2016)). Such features help segregate the socializing class in subtask A. For subtask B; we exploit the relation between what type of information the user wants to ask (QType) and what type of information is provided in the answer (NET). To capture this, we extract the type of all named-entity mentions in the answer. We consider "person", "organization", "location" and "quantity" as possible NE tags extracted using spacy[5].

5.5 External Evidence

In subtask B, the verification of an answer requires external evidence to conclude about its veracity. We extract external evidence by formulating a search-query from the question and answer followed by a web search [6] of this search-query. For each of the obtained search results, we compute its similarity with the question and answer respectively. These similarity scores are then used as features to a classifier.

Search-Query Formulation In order to search the web for relevant evidence, we formulate a search-query based on the question and answer. We extract query-sentence from the question posted by a user and append "Qatar" if neither 'Doha' nor 'Qatar' is present in it.

Further, to incorporate relevant information from the answer into this search-query, we find the answer sentence that has the highest similarity with the query-sentence. From this top-ranked sentence, we extract up to 7 keywords based on named entities, noun-chunks[5] and unigrams sorted by tf-idf scores, where named entities and noun-chunks are given high priority. Query-sentence combined with keywords from the answer is used as search-query.

Search Results We collect search results (snippets) from reputed sources (e.g., news, government websites, official sites of companies) (Mihaylova et al., 2018) for search-query formulated as above. Since the search-query may not always be perfect, we also obtain search results by drop-

[4]https://www.clips.uantwerpen.be/pages/pattern-en

[5]https://spacy.io/usage/linguistic-features

[6]Bing web search API

Question	Answer	Evidence	Source	Class
how cold is doha during winter?	**8-10 degree** i guess	Over the course of the year, the temperature typically varies from **57 F (14 C) to 107 F (42C)** and is rarely below 51F or above 112F	Weatherspark	False
any private beaches in Qatar?	While Going to Shammal after 40 KM from Doha you will find **Al Ghariyah**	Qatar is a peninsula....This is a list of beaches in Qatar. Contents.**Al Ghariyah beach** is located 80 km north of Doha.	Wikipedia	True

Table 3: Example of external evidence collected in Subtask B

ping a few keywords from the search-query. From all the obtained search results, we select snippets that are most relevant to the question and the answer. Table 3 illustrates the external evidence retrieved for two question-answer pairs.

Similarity based Features For each question-answer pair, we compute their similarity with external search results obtained above. We use three similarity metrics: containment of unigrams, bigrams and trigrams (Lyon et al., 2001), cosine similarity of USE embedding and cosine similarity of tf-idf representation. For each metric, we compute the similarity of the snippet with: question, answer, query-sentence + top-answer sentence and all of them together. We then take the average and maximum over similarity scores for all the search results.

5.6 Forum-Level Features

These features capture the relevance of an answer with respect to a question as well as to other answers. An answer which contains information similar to that specified in other answers is more likely to be relevant and trustworthy. Thus, we consider the similarity of the answer with the question as well as its similarity with other answers in that thread. Here, also we consider all three similarity metrics mentioned before.

6 Experiments and Evaluation

6.1 Setting and Evaluation Metrics

We now utilize all features as portrayed in Figure 1 for subtask A and Figure 2 for subtask B. We train two separate SVM classifiers (Burges, 1998) on respective features for 3-class classification for both the subtasks. We use 10-fold cross validation for hyper-parameter tuning of SVM based on which, we choose "linear" kernel with C=0.5 (regularization parameter) for all the demonstrated experiments. All the results are reported on the test data with accuracy, recall, and F1 measure as evaluation metrics.

Additionally, we calculate Mean Average Precision (MAP) for subtask B, where the 'True' instances are considered relevant examples (in the context of Information Retrieval). MAP measures the capability of the system to predict 'True' instances with higher confidence.

6.2 Results for Subtask A

Table 4 shows the performance of the proposed system (PS). From the results, we can observe that our PS (excluding syntactic features) achieves an impressive performance with accuracy of 84.12% and 72.17% F1. Our submission (with all the features in PS + (POS and QType)) achieved similar performance and ranked 3rd on the leaderboard (83% acc., 71% F1) with only a marginal difference with respect to the first (84% acc., 72% F1) and second-ranked (83% acc., 72% F1) systems. To push our system's performance even further, we experimented in the post-evaluation phase and achieved 88.10% accuracy and 77.37% F1, highest on the post-evaluation leaderboard [7]. This current best solution leverages QType features, extensive data augmentation using bagging technique and excludes writing style features.

In order to appraise the importance of each feature, we conducted an ablation study by analyzing individual features and their combinations. It can be followed from the results that the semantic embedding contributes most to the performance of the system. However, embeddings derived using USE perform better than glove embeddings. This difference is possibly due to the failure of glove-based embedding in capturing word-order and long-range dependencies.

The second most important contributor is the data augmentation approach which resulted in notable accuracy gains (3.71% improvement). As expected, it allows the system to generalize better and ameliorates the issue of class imbalance. Following it is the query-sentence extraction approach with $\sim$1% accuracy enhancement. The

[7]As reported on 23/2/2019

Feature	Type	Acc	F1	Rec
PS+ Syntactic	POS	82.30	69.23	71.26
	Cat	83.77	71.61	73.52
	QType	84.67	73.00	74.75
PS	USE	**84.12**	**72.17**	**73.90**
	Glove	78.55	64.11	65.79
PS-WS		86.36	75.64	76.96
PS-Ling		83.49	72.09	74.94
PS-QSent		83.21	71.72	74.71
PS-DA		80.41	67.53	71.17
PS-Sem		70.97	52.95	54.92
Submission	Official	83.14	70.89	72.82
Best	Ensemble	**88.10**	**77.37**	**77.96**

Feature	Type	Acc	F1	Rec	MAP
PS+ Syntactic	POS				
	Cat				
	QType+NE	77.63	42.46	42.74	30.00
PS (Best)	USE	76.56	42.65	45.12	25.00
	Glove	**77.85**	**43.90**	**44.31**	**58.33**
PS-Ext		78.71	45.75	46.08	25.00
PS-Forum		75.48	41.30	42.25	62.50
PS-Sem		73.98	42.08	45.42	23.81
PS-Reputed		72.90	37.93	39.09	29.17
PS-DA		67.74	37.21	40.28	37.50
Submission	Official	69.00	37.44	40.25	33.33
Baseline	Majority	**83.01**	**28.47**	**33.33**	**15.55**

Table 4: Experimental Results. Subtask A (Left) and Subtask B(Right)
PS: Proposed System, WS: Writing Style, Ling: Linguistic, Qsent: Query Sentence, DA: Data Augmentation, Sem: Semantic Embedding, Cat: Category, Ext: External Evidence, Forum: Forum level evidence, NE: Named Entity from answer

performance is in line with the expectations as query-sentences are sufficient in capturing the essence of user question. QType and linguistic cues help improve the performance further.

However, we notice that the performance improves by excluding the writing style features. The possible reason for this observation can be the absence of such features in the test data. In the training and development data-sets, the presence/absence of these features was a distinguishing factor among classes (see Table 2) which made them worth considering.

6.3 Results for Subtask B

Table 4 shows the performance of PS with the ablation study. Our PS (also our best[7]) achieves an accuracy of 77.85% with 58.33 MAP (highest among all the participating systems). From the ablation study, we observe that although removal of external evidence results in slight accuracy gains (0.86%), it causes a drastic reduction in MAP score (33.33 points). This signifies the importance of external evidence as these features enable the system to make better predictions for the true/false classes.

We also conducted a majority baseline experiment where all the samples are labeled as "non-factual." This experiment resulted in the best performance on the leaderboard with 83% accuracy and very poor MAP. This illustrates that the test data has a majority of non-factual instances. Thus, measuring the performance of any system solely on accuracy for this problem is not fair.

As it can be inferred from the ablation study, among all the features, reputed source based search-results selection (contributing 4.95% acc. gain) and forum-level features (contributing 2.37% acc. gain) are the most important. Reputed source selection helps in relying on only trusted sources for external evidence selection and hence make better predictions for true/false classes. Forum level features help in distinguishing among non-factual versus true/false samples.

Further, data augmentation in subtask B results in significant performance gains of 10.11% accuracy. It helps the system learn about the characteristics of "bad answers" which are not present in the training data and hence enables the system to generalize better on the test data. For this subtask as well, semantic embedding plays a vital role in capturing the essence of the question-answer pair, contributing 3.87% gain in accuracy.

7 Conclusion

In this work, we have described our system for Semeval-2019 Task 8 on Fact-checking in cQA Forums. Our system leverages data augmentation and integrates knowledge from various aspects, such as the semantics, linguistics, syntax and writing style along with complementary information from trustworthy external sources and QLF.

Our submission was ranked third in Subtask A with marginal performance differences compared to the best-ranked systems. Our current best solution is ranked first on the leaderboard with 88% accuracy[7]. In subtask B, our current best solution is ranked 2nd, with 58.33% MAP score, highest among all participating systems[7].

However, none of the participating systems could beat the majority baseline for subtask B in terms of accuracy, which signifies that we are still far from solving this task to its entirety with a decent performance. Thus, there remains a lot of potential in this research direction.

References

Christopher JC Burges. 1998. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Joan B Hooper. 1974. *On assertive predicates*. Indiana University Linguistics Club.

Ken Hyland. 2018. *Metadiscourse: Exploring interaction in writing*. Bloomsbury Publishing.

Lauri Karttunen. 1971. Implicative verbs. *Language*, pages 340–358.

Caroline Lyon, James Malcolm, and Bob Dickerson. 2001. Detecting short passages of similar text in large document collections. In *Proceedings of the*.

Harish Tayyar Madabushi and Mark Lee. 2016. High accuracy rule-based question classification using question syntax and semantics. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1220–1230.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Tsvetomila Mihaylova, Preslav Nakov, Lluis Marquez, Alberto Barron-Cedeno, Mitra Mohtarami, Georgi Karadzhov, and James Glass. 2018. Fact checking in community forums. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. Semeval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 27–48.

Preslav Nakov, Lluís Màrquez, Walid Magdy, Alessandro Moschitti, Jim Glass, and Bilal Randeree. 2015. Semeval-2015 task 3: Answer selection in community question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 269–281.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. SemEval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Ellen Riloff and Janyce Wiebe. 2003. Learning extraction patterns for subjective expressions. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*.

Veronika Vincze. 2013. Weasels, hedges and peacocks: Discourse-level uncertainty in wikipedia articles.

Jeffrey S Vitter. 1985. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57.

TMLab SRPOL at SemEval-2019 Task 8: Fact Checking in Community Question Answering Forums

Piotr Niewiński, Aleksander Wawer, Maria Pszona, Maria Janicka
Samsung R&D Institute Poland
pl. Europejski 1
00-844 Warsaw, Poland
{p.niewinski, a.wawer, m.pszona, m.janicka}@samsung.com

Abstract

The article describes our submission to SemEval 2019 Task 8 on Fact-Checking in Community Forums. The systems under discussion participated in Subtask A: decide whether a question asks for factual information, opinion/advice or is just socializing. Our primary submission was ranked as the second one among all participants in the official evaluation phase. The article presents our primary solution: Deeply Regularized Residual Neural Network (DRR NN) with Universal Sentence Encoder embeddings. This is followed by a description of two contrastive solutions based on ensemble methods.

1 Introduction

Community question answering forums are good platforms for knowledge sharing; hence, they are widely used sources of information. The growing popularity of such knowledge exchange leads to a growing need to automate the process of verifying the post quality. The first step, often overlooked, is to categorize each question and establish what kind of information the user seeks.

Question classification has been mainly used to support question answering systems. Two main method types have been proposed in the literature: (1) rule-based approaches with linguistic features (Tomuro, 2004; Huang et al., 2008; Silva et al., 2011), and (2) machine learning approaches (Zhang and Lee, 2003; Metzler and Croft, 2005). These methods are rather simple, due to the fact that question classification is often just a preprocessing step in a larger task. However, we can observe some recent advances in this area, such as ULMFiT (Howard and Ruder, 2018), which achieves state-of-the-art performance on the TREC dataset (Voorhees and Tice, 1999).

The present article describes our systems submitted to the SemEval 2019 competition Task 8 subtask A on question classification. The competition data set consisted of QatarLiving forum questions classified as FACTUAL, OPINION or SOCIALIZING. The training data contained only 1,118 questions. Moreover, according to our evaluation, human-level accuracy on this data set was about 0.75, which was relatively low. Therefore, we approached the task as a challenging classification problem.

The article is structured as follows. Section 2 presents our experiments with preprocessing methods. Section 3 describes our official submission, where we propose an architecture utilizing several regularization methods to address the problem of the small data set. For comparative purposes, section 4 presents two ensemble models as contrastive examples. Section 5 provides the results achieved by the models. Lastly, section 6 concludes the discussion.

2 Data Preprocessing

We tested a few simple text preprocessing setups. Unfortunately, none of them helped the models achieve improved results. Hence, they are here presented as negative results, and for reference.

First, all emojis were removed from the text, and all URLs were replaced with the string 'url link'. Next, all dates and hours were replaced with 'date' and 'hour', respectively. Ordinal numbers – i.e. 1^{st}, 2^{nd}, 5^{th} etc. – were replaced with 'nth', while the remaining numbers were substituted wtih 'num'. All of these sequences were found using regular expressions. Furthermore, if most of the letters were uppercase, the whole text was lowercased.

Second, some of the forum-specific jargon was replaced with more generally used terms. This was achieved by an internally prepared dictionary that translated 'qar' into 'Qatar currency', 'qling' into ‘browsing Qatar forum’, ‘ql’ into ‘Qatar forum’, ‘villagio’ to ‘Qatar shopping center’, etc. Additionally, it helped us to correct common spelling errors, such as ‘doha’ for ‘Doha’ and ‘qatar’ for ‘Qatar’. Finally, spelling correction was performed by a custom character-based CNN language model. This way, we hoped to obtain a better representation of texts when embedded into vectors.

However, the experiments showed that none of these methods brought significant improvement in classification accuracy. It seemed that noise removal, combined with text normalization, deprived the data of significant features and information which carried crucial meaning for preparing text embeddings. Therefore, we finally did not perform any preprocessing and worked on raw question subjects and body text.

3 Primary Submission

3.1 Features

The feature space for the models was created by combining three different sources of information:

1. *Universal Sentence Encoder* – The concatenated post subject and body text were embedded with the Universal Sentence Encoder (USE) (Cer et al., 2018) to create a 512-dimensional vector representation.

2. *fastText embeddings* – The concatenated post subject and body text were tokenized with the Spacy library and embedded on the word level with 300-dimensional fastText vectors. Then, the vectors were averaged on the sequence dimension.

3. *Category statistics* – For each QL post category, the ratio of the FACTUAL, OPINION and SOCIALIZING labels was calculated. The three numbers were normalized, forming a 3-dimensional vector.

The three subfeature vectors were concatenated to produce an 815-dimensional vector for each question.

3.2 Model Architecture

We proposed the Deeply Regularized Residual Neutral Network architecture, shown in Figure 1.

The model took as its input the 815-dimensional vector of floats (concatenated USE embeddings,

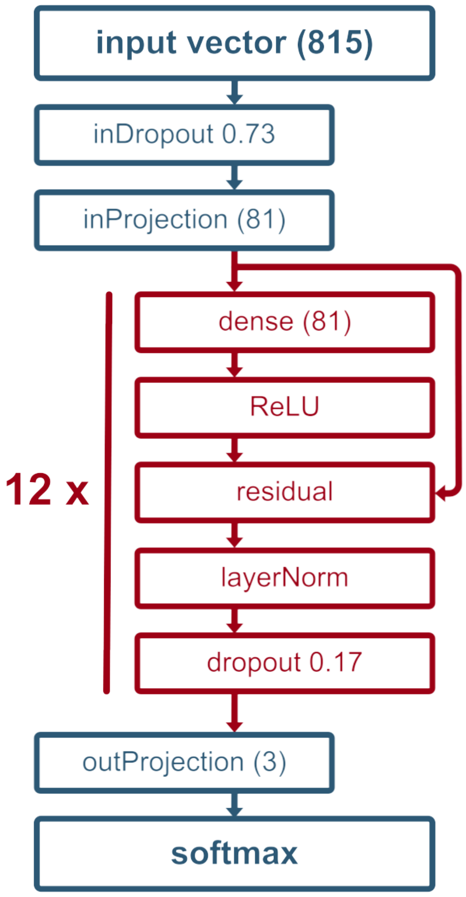


Figure 1: The architecture of DRR NN (primary submission).

fastText embeddings averaged, and category statistics). During the training, a large dropout of 0.73 was applied to the input vector.

The core of the model was a deep subnetwork built of 12 stacked blocks. Each block contained an 81-dimensional dense layer followed by ReLU activation, residual connection, layer normalization and 0.17 dropout. Finally the output of the last block was projected with a dense layer into a 3-dim logits vector.

The model was trained with the Adam optimizer, at a 6e-3 learning rate, and with 500-epoch linear warmup. We used softmax cross entropy loss with 0.14 of L2 penalty regularization.

All model hyperparameters were optimized with a randomized search algorithm and 5-fold cross-validation over the training data set. The final model size was 148K learnable variables.

3.3 Model Training

The main idea behind the advanced training procedure was to split the training data into a bigger learning part and a smaller validation part. The loss was minimized on the learning part until the accuracy on the validation part began to increase.

Generally, model performance depends on many factors, such as training efficiency, model architecture, optimization algorithms, etc. At the same time, it is affected by sample distribution between learning and validation parts.

In order to aggregate more knowledge from the training data, we used 5-fold cross-validation splits. We prepared 4 such splits using different random seeds. This procedure gave us a total of 20 different pairs of learning/validation sets.

We set the maximum number of epochs to 700. The model was validated after each training epoch and saved until its classification accuracy improved. Usually, the accuracy was improving for about 300-600 epochs. For the final prediction, we used the argmax of the summarized softmax of 20 models:

$$\arg\max \sum_{k=1}^{20} \mathrm{softmax}(logits_k).$$

4 Contrastive Submissions

For the contrastive submissions, our overall idea was to utilize multiple models that were as varied as possible, and combine their outputs.

In the first step, we used the following systems to obtain label probabilities for each question:

- *ELMO* (Peters et al., 2018) – a deep, contextualized word representation to obtain sentence representation, followed by a neural network of two dense layers. We arrived at the following architecture and hyper-parameters during the optimization: a dense layer of 48 neurons (dropout 0.5), followed by a second dense layer of 10 neurons (dropout 0.5). When tested on the training data in cross-validation, this solution alone achieved a micro-accuracy of 0.72.

- *BERT* (Devlin et al., 2018) – a deep, bidirectional transformer model with sequence classification layers on the top. The BERT language model was pre-trained, so only the sequence classifier was initialized and trained on the SemEval data. We used the PyTorch implementation of the case-insensitive 'base' version[1] with the optimal number of epochs (10) determined on the development set. When tested on the training data in cross-validation, this solution alone achieved a micro-accuracy of 0.717.

- *Bag-of-words* – a machine learning solution based on character n-gram vectorization with TF-IDF weighting and a linear kernel SVM classifier. We used the implementation from the scikit-learn package (Pedregosa et al., 2011). When tested on the training data in cross-validation, this solution alone achieved a micro-accuracy of 0.699.

In the second step, we prepared two different ensemble models combining the probability outputs from *ELMO*, *BERT*, *Machine Learning* and *DRR NN*.

The first contrastive submission (**Contrastive-1**) used the SVM classifier with linear kernel. The second contrastive submission (**Contrastive-2**) was designed as a bagging classifier of 10 estimators, each a voting ensemble of logistic regression, random forest and SVM with linear kernel.

5 Results

Table 1 contains the results of the evaluation on the official test set. The primary submission and both contrastive submissions were presented during the official phase of the contest. After the official competition, we tested additional solutions. Surprisingly, we achieved the best results with the SVM classifier (RBF kernel) on the USE embedding (**Post-evaluation**).

Model	**Accuracy**	**F1**	**AvgRec**
DRR NN (Primary)	0.83	0.72	0.76
Contrastive-1	0.83	0.72	0.76
Contrastive-2	0.81	0.69	0.73
Post-evaluation	0.87	0.77	0.78

Table 1: Official results of our submissions on the test set.

6 Conclusions

According to the experiments, and as reflected in the results on the test set, the best performing system was DRR NN based on the Universal Sentence Encoder. We attributed its good performance

[1] https://github.com/huggingface/pytorch-pretrained-BERT

on the small data set to the deep regularization and the advanced training procedure. However, the SVM classifier performed even better, probably thanks to its overfitting resistance (Xu et al., 2009).

Additionally, we tested several approaches, including the usual high performers, such as BERT or ELMO, and the ensemble systems. None of them was able to outperform our primary submission. We attribute such behaviour to data over-fitting and lack of ability to extract higher-level dependencies from the provided samples.

Some influence on the results could have been exerted by the significantly differing distributions of post categories among the train, dev and test sets. For example, while more than 30% of all questions from the test set belonged to the 'Visas and permits' category, only 8% from the train set and 5% from the dev set fall into the same category.

Linear SVM with the USE embeddings reached an accuracy of 0.84 on the dev set and 0.86 on the test set. Surprisingly, with a different set of parameters, we achieved 0.87 accuracy on the test set, and only 0.81 accuracy on the dev set.

References

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Zhiheng Huang, Marcus Thint, and Zengchang Qin. 2008. Question classification using head words and their hypernyms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 927–936. Association for Computational Linguistics.

Donald Metzler and W Bruce Croft. 2005. Analysis of statistical question classification for fact-based questions. *Information Retrieval*, 8(3):481–504.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Joao Silva, Luísa Coheur, Ana Cristina Mendes, and Andreas Wichert. 2011. From symbolic to subsymbolic information in question classification. *Artificial Intelligence Review*, 35(2):137–154.

Noriko Tomuro. 2004. Question terminology and representation for question type classification. *Terminology. International Journal of Theoretical and Applied Issues in Specialized Communication*, 10(1):153–168.

Ellen M Voorhees and Dawn M Tice. 1999. The trec-8 question answering track evaluation. In *TREC*, volume 1999, page 82. Citeseer.

Huan Xu, Constantine Caramanis, and Shie Mannor. 2009. Robustness and regularization of support vector machines. *Journal of Machine Learning Research*, 10(Jul):1485–1510.

Dell Zhang and Wee Sun Lee. 2003. Question classification using support vector machines. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 26–32. ACM.

TueFact at SemEval 2019 Task 8: Fact checking in community question answering forums: context matters

Réka Juhász
University of Tübingen
Wilhelmstr. 19-23
72074 Tübingen, Germany
reka.juhasz@student.uni-tuebingen.de

Franziska Barbara Linnenschmidt
University of Tübingen
Wilhelmstr. 19-23
72074 Tübingen, Germany
franziska-barbara.linnenschmidt@student.uni-tuebingen.de

Teslin Roys
University of Tübingen
Wilhelmstr. 19-23
72074 Tübingen, Germany
teslin.roys@student.uni-tuebingen.de

Abstract

The SemEval 2019 Task 8 on Fact-Checking in community question answering forums aimed to classify questions into categories and verify the correctness of answers given on the QatarLiving public forum. The task was divided into two subtasks: the first classifying the question, the second the answers. The TueFact system described in this paper used different approaches for the two subtasks. Subtask A makes use of word vectors based on a bag-of-word-ngram model using up to trigrams. Predictions are done using multi-class logistic regression. The official SemEval result lists an accuracy of 0.60. Subtask B uses vectorized character n-grams up to trigrams instead. Predictions are done using a LSTM model and achieved an accuracy of 0.53 on the final SemEval Task 8 evaluation set. In a comparison of contextual inputs to subtask B, it was determined that more contextual data improved results, but only up to a point.

1 Introduction

The SemEval 2019 Task 8 on Fact-Checking gave us the opportunity to develop a system that evaluates the factual content of questions and answers in the field of community question answering forums. This popular niche on the internet provides helpful information for specific interests, such as information on life in Qatar or elsewhere in the world, coding help on Stack Overflow, or answers to a wide range of questions on Quora, Reddit/r/Ask or Yahoo! Answers. Often other resources are not at hand or misleading, and it proves difficult to find what one is looking for amid non-relevant questions, let alone be sure the answers found are correct. A system that can, with some degree of accuracy, pick out the factual questions and then predict the correctness of the given answers, is a means to ensure quality in community question answering forums. There may also be many further applications in information retrieval – ordering search results based on estimated factuality in a web query or even identifying truthful answers in automatic question answering systems.

The task was divided into two subtasks. While Subtask A required a classification of the questions into three distinct categories "Factual", "Socializing", and "Opinion", Subtask B took the subset of the "Factual" questions to classify them according to either "True", "False" or "NonFactual" in terms of the actual answer. Similar tasks were already part of SemEval 2015 (Nakov et al., 2015) and SemEval 2016 (Nakov et al., 2016).

The Tuefact system follows this division, even going as far as using different pre-processing to accommodate for the different needs. While the question classification is done with a bag of word approach using word trigrams, the answer classification uses character trigrams. The models used to make predictions are logistic regression for Subtask A and a long short-term memory model (LSTM) (Hochreiter and Schmidhuber, 1997) for Subtask B.

The following section describes the data provided for the tasks. The next two sections describe the two components of our language independent system in detail together with a brief discussion of failed approaches and changes that lead to improvements. In the last two sections we discuss further work to be done on the TueFact system and our conclusion about the current version of it.

2 The data

The pre-annotated data from the QatarLiving forum was provided in XML format. It was split into two files: one for the question classification, the other for the answer classification.

The data for Subtask A comprised a total of

168 questions. Each question contained a subject line, i.e. a summary of the question, the detailed question, and all answers given to the question. The meta information contained amongst others the date, user name, and id. The questions were annotated into the three categories “Factual”, “Opinion”, and “Socializing”. 33 of the questions were labeled as “Factual”, 73 as “Opinion”, and 62 as “Socializing”. The longest question was 98 words long, the shortest only 5, the average length of questions was 41.1 words.

The data provided for Subtask B contained a total of 95 questions labeled “Factual”. The meta information was the same. Of the 356 given answers 128 were labeled as “True”, 102 as “False”, and 126 as “NonFactual”. The longest answer was 195 words long, the shortest consisted of only one word, the average length of answers was 31.5 words.

For further information about the data please refer to the task description paper from Mihaylova et. al (2019). No special data pre-processing steps were used in preparation for either subtask – no noticeable performance gains were observed when stripping accents, punctuation symbols, or lowercasing.

3 Question classification

The goal of this subtask was to classify the questions posted on the QatarLiving forum,[1] a community question answering forum, as either: 1. “Factual”, meaning that it asks about something specific, and can receive a correct or incorrect answer, 2. “Opinion”, in which the answers cannot be right or wrong, as it does not ask for objective facts but the personal input of the answering people, and 3. “Socializing”, where the goal of posting the question was not seeking information at all, but rather looking for online communication.

[1]https://www.qatarliving.com/forum

3.1 Approach

We approached this task as a multi-class learning problem, instead of first dividing the questions into “Factual” and “NonFactual”, and then further dividing the “NonFactual” questions into “Opinion” and “Socializing”. As baseline model we decided to use multi-class logistic regression based on character bigrams, and only the subject line of the questions as input. Our reasoning for the use of this baseline model was to see how such a basic model would perform, and where we could take it from there. It achieved an accuracy of just over 50% on our development set, a randomly split quarter from the data set provided.

The current model uses a multi-class logistic regression model with a limited-memory Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm (Saputro and Widyaningsih, 2017) solver to predict the labels. The labels were encoded using the scikit-learn library (Pedregosa et al., 2011) and the questions vectorized from word uni-, bi- and trigrams. We used raw counts for all word-grams, and did not add weighting. We further decided not to use a language dictionary as external feature, in order to maintain language independence.

3.2 Results

For means of evaluating our models, and as the data set was small, we have randomly split the training data and used one quarter of it as a development set. The development set was then not only used to evaluate our model in terms of accurcay, but also to compare the predicted to the actual labels. Our system was best at correctly predicting “Opinion” in an average of 10 cases compared to an average of four misclassified cases with no clear tendency of misclassifying it as either “Factual” or “Socializing”. False predictions were equally often made in the classification of the labels “Socializing” and “Factual”, which suggests further improvement possibilities.

Best training results on the development set were achieved at an accuracy of 0.714. The official SemEval result lists an accuracy of 0.60.

	No CC	Limited CC	Subject + IDs Only	Full CC
Accuracy	61.5	67.12	79.94	79.92

Table 1: Averaged 5-fold cross-validated accuracy on the development set for the character-based embeddings variation.

4 Answer classification

In this subtask, answers to the questions from the first task were to be classified into "Factual" and "Non-Factual", and "Factual" items further classified into "True" or "False". Unlike in the first task, there is no distinction made between types of non-factual comments (e.g. socializing or opinion).

4.1 Approach

Our approach to this task was to treat the problem as a multiple-class learning problem with three categories: "Non-Factual", "True" and "False". Variations of the model which split it into two sequential learning problems – fact or non-fact, then true or false – showed no significant difference.

The basic task B model consists of a LSTM network architecture (Hochreiter and Schmidhuber, 1997) with an embedding layer, two hidden layers of 100 nodes followed by a softmax activation layer to permit multiple classification. Embeddings were trained using the top 400 uni-, bi- and tri-gram features by frequency in the data. In training, we used categorical cross-entropy as a loss function. In all model variations, we trained for 200 epochs with a batch size of 128 and used L2 weight regularization with a factor of 0.012.

4.2 Variations and results

We examined three main variations of input to the model for answer classification: no comment chain (CC), limited comment chain, subject and comment identifier only and full comment-chain (see table 1). In no CC, we included only the comment itself being classified. In limited CC, we included the comment and all previous comments in the chain. In the subject plus comment identifier variation, we included the question subject heading, the comment text and commenter identifier. In the full CC variation we included the subject heading and the entirety of all comments in the chain. In tables 1 and 2, we abbreviate the same way.

Next, we experimented with word-level versus character-level embeddings (see table 2). In the end, unfortunately, our best results for task B on the evaluation set (also with the subject and identifiers only variation) were less encouraging at 53 percent.

5 Future work

Due to time constraints, several improvements to the models for both tasks weren't completed in time for the final evaluation. For the question classification task, we aimed to experiment with word- or character-embeddings instead of only a bag-of-n-grams approach in order to enable work on a multi-channel convolutional neural network model, which is also being pursued further. In some NLP tasks, CNNs have been shown to outperform not only traditional machine learning models but recurrent neural networks as well, and this may also be the case here (Wu et al., 2016).

For the answer classification task, work is underway on a model variation which tags comments based on the proportion of their content that can be found on multiple websites from a web query consisting of the question subject line.

6 Conclusions and analysis

For Subtask A, we considered two approaches: simple logistic regression, and a convolutional neural network approach. The initial success of the logistic regression approach on the development data suggests a 'simplicity first' strategy is sensible in this case, but its mediocre performance on the evaluation data indicates it is not especially robust.

In Subtask B, we examined only a single approach using a LSTM trained with embeddings, but with a number of variations. Variations including more of the comment chain as input were more successful than using single-comment answers as input, however we found little difference between including the entire comment chain and including only the subject question and comment identifiers. We suspect this is due to the fact that all answers include each other at least once in the full-chain variation, so it provides less to distinguish the answers from one another. Further, all vari-

	No CC/WB	Limited CC/WB	Full CC/WB
Accuracy	59.87	66.97	79.14

Table 2: Averaged 5-fold cross-validated accuracy on the development set for the word-based embeddings variation.

ations using character-based embeddings slightly outperformed model variations employing word-based embeddings (1 to 5 percent). This isn't entirely unexpected – solely character embedding based models performed very well in the SemEval 2018 Irony Detection task (Van Hee et al., 2018) as well, which bore similarities. One hypothesis for this result is that with small input sizes (such as tweets or forum posts) word-based embeddings may distinguish fewer distinctions, but it may also simply be that there is no significant performance difference. In which case, character- embeddings should be preferred as they do not require a maintenance of a large dictionary.

Overall, the TueFact system is an acceptable baseline and a solid starting point for further work in the direction of fact checking in community question answering forums. But perhaps of greatest interest are our comparative results under different input. The significance of input choice on performance is highlighted: our results show that while inclusion of more context can certainly be useful, the intuition that more data will always improve performance in this task should not be taken as a given.

References

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Tsvetomila Mihaylova, Georgi Karadzhov, Atanasova Pepa, Ramy Baly, Mitra Mohtarami, and Preslav Nakov. 2019. SemEval-2019 task 8: Fact checking in community question answering forums. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '19, Minneapolis, MN, USA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Preslav Nakov, Lluís Màrquez, Walid Magdy, Alessandro Moschitti, Jim Glass, and Bilal Randeree. 2015. Semeval-2015 task 3: Answer selection in community question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015*, pages 269–281.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. Semeval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 525–545.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Dewi Retno Sari Saputro and Purnami Widyaningsih. 2017. Limited memory broyden-fletcher-goldfarb-shanno (l-bfgs) method for the parameter estimation on geographically weighted ordinal logistic regression model (gwolr). In *AIP Conference Proceedings*, volume 1868, page 040009. AIP Publishing.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. Semeval-2018 task 3: Irony detection in english tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 39–50.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

YNU-HPCC at SemEval-2019 Task 8: Using A LSTM-Attention Model for Fact-Checking in Community Forums

Peng Liu, Jin Wang and **Xuejie Zhang**
School of Information Science and Engineering
Yunnan University
Kunming, P.R. China
`Contact:xjzhang@ynu.edu.cn`

Abstract

The objective of the task, Fact-Checking in Community Forums , is to determine whether an answer to a factual question is true, false, or whether it even constitutes a proper answer. In this paper, we propose a system that uses a long short-term memory with attention mechanism (LSTM-Attention) model to complete the task. The LSTM-Attention model uses two LSTM(Long Short-Term Memory) to extract the features of the question and answer pair. Then, each of the features is sequentially composed using the Attention mechanism, concatenating the two vectors into one. Finally, the concatenated vector is used as input for the MLP (Multi-Layer Perceptron) and the MLP's output layer uses the softmax function to classify the provided answers into three categories. This model is capable of extracting the features of the question and answer pair well. The results show that the proposed system outperforms the baseline algorithm.

1 Introduction

Many questions pertaining to various fields are posted to QA forums by users every day, where they collect answers. However, the answers do not always address the question asked. Indeed, in some cases, the answer has nothing to do with the question. There are several reasons why this is the case. For example, the responder could have misunderstood the question and so provided a wrong answer. Most QA forums have little control over the quality of the answers posted. Moreover, in our dynamic world, the true answer was true in the past, but it may be false now . Figure 1 presents an example from the Qatar Living forum. In this case, all three answers could be considered to be good since they formally answer the question. Nevertheless, a1 contains false information, whereas a2 and a3 are correct, as can be established from the official government website.

q**:** I have heard its not possible to extend visit visa more than 6 months? Can U please answer me.. Thankzzz...

a_1**:** Maximum period is 9 Months....

a_2**:** 6 months maximum

a_3**:** This has been anwered in QL so many times. Please do search for information regarding this. BTW answer is 6 months.

Figure 1: An example from the Qatar Living forum

In this study, we aim to solve the problem of detecting true factual information in online forums. Given a question requesting factual information, the goal is to classify the provided answers into the following categories.

(i) Factual - True: The answer is true and can be proved by cross referencing with an external resource.

(ii) Factual - False: The answer gives a factual response, but it is either false, partially false, or the responder is uncertain about their response.

(iii) Non-Factual: The answer does not provide factual information relevant to the question; it is either an opinion or an advice that cannot be verified.

To the best of our knowledge, various approaches have been proposed for the purposes of fact-checking in community forums (Mihaylova et al., 2018), such as long short-term memory (Gers et al., 2000) .

In this paper, we provide an LSTM-Attention model for fact-checking in community question answering forums. In our approach, we use pretrained word vectors for word embedding. The LSTM layer is used to extract features from the question and answer sentences. Finally, these features are used by the Attention Mechanism (Vaswani et al., 2017) with a focus on extracting useful information from the features that are significantly relevant to the current output.

The remainder of this paper is organized as fol-

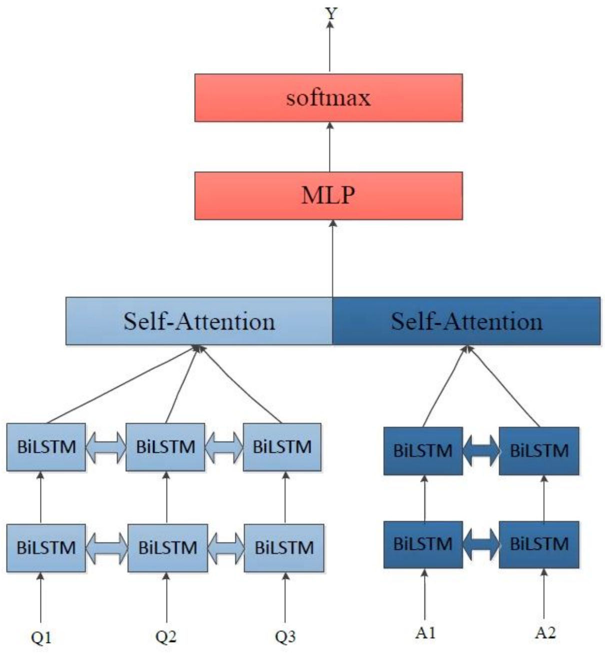


Figure 2: LSTM-Attention Model

lows. In section 2, we described the LSTM, Attention model, and their combination. Section 3 summarizes the comparative results of the proposed model against the baseline algorithm. Section 4 concludes the paper.

2 LSTM-Attention Model for Fact-Checking

Figure 2 shows the architecture of our model. First, a sentence is transformed into a feature matrix. The feature matrix is then passed into the LSTM to extract salient features.

A simple tokenizer is used to transform each sentence into an array of tokens, which constitute the input to the model. This is then mapped into a feature matrix or sentence matrix by an embedding layer. The n-gram features are extracted when the feature matrix passes through the LSTM, and the output of the LSTM is passed into the Self-Attention layer. This layer composes the useful features to output the final regression results by means of a linear decoder.

2.1 Embedding Layer

Vectors encoded using the one-hot method have large dimensions and are sparse. Suppose we encounter a 2,000-word dictionary in natural language processing (NLP). When the one-hot method is used for coding, each word will be represented by a vector containing 2,000 integers. If the dictionary is larger, this method will be very inefficient.

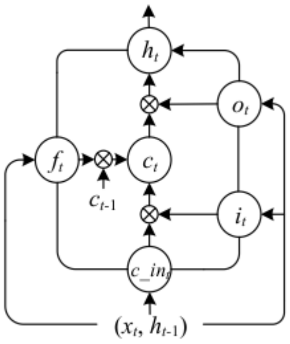


Figure 3: LSTM

The one-hot-vector method has many defects when used for word encoding. One is that it has too much redundancy; the other is that the dimension of the vector is too high. The vector will have as many dimensions as there are words, which will increase the computational complexity. Word-embedding Mikolov et al. (2013) transforms an original high-dimensional redundant vector into a low-latitude vector with strong information content. No matter how many words there are, the converted vector generally has only 256 dimensions to 1024 dimensions.

The embedding layer is the first layer of the model. Each sentence is regarded as a sequence of word tokens $t_1, t_2...t_n$, where n is the length of the token vector.

2.2 Long Short-Term Memory

In theory, RNN Tsoi and Back (1994) should be able to handle such long-term dependency. We can pick and choose the parameters carefully to solve the most elementary form of this type of problem (Le et al., 2015). However, in practice, RNN is not able to learn this knowledge successfully. Therefore, the LSTM was designed to solve the problem of long-term dependency. In practice, the LSTM excels at dealing with long-term dependency information rather than the ability to acquire it at great cost. RNN has a chain of repeating neural network modules. In standard RNN, the repeating module has a very simple structure. LSTM has the same structure, but the structure of repeating modules is more complex. This is different from that of the single neural network layer. Figure 3 shows the detailed structure of an LSTM. The LSTM cal-

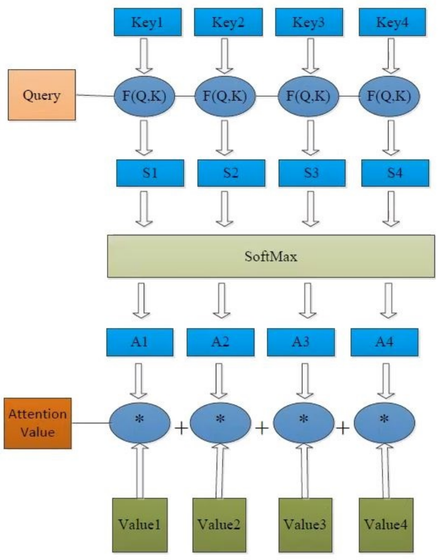


Figure 4: Attention

culates hidden states H_t and outputs C_t using the following equations.

• Gates:

$$\begin{aligned} i_t &= \sigma\left(W_{xi}x_t + W_{xi}h_{t-1} + b_i\right) \\ f_t &= \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + b_f\right) \\ o_t &= \sigma\left(W_{xo}x_t + W_{xo}h_{t-1} + b_o\right) \end{aligned} \quad (1)$$

• Input transformation:

$$c\_in_t = tanh\left(W_{xi}x_t + W_{xi}h_{t-1} + b_{in}\right) \quad (2)$$

• State update:

$$\begin{aligned} c_t &= f_t \otimes c_{t-1} + i_t \otimes c\_in_t \\ h_t &= o_t \otimes tanh\left(c_t\right) \end{aligned} \quad (3)$$

Here, x_t is the input vector; c_t is the cell state vector; W and b are layer parameters; f_t , i_t , and o_t are gate vectors; and σ is the sigmoid function. Note that $\otimes$ denotes the Hadamard product. Bidirectional LSTM comprises a forward LSTM and a reverse LSTM. It captures context feature information very well as compared to LSTM. Therefore, bidirectional LSTM usually performs better than LSTM and we use it to process the sequences. Among the many hidden layers of deep neural networks, the earlier layers learn simple low-level features, and later layers combine simple features to predict more complex things. Therefore, we use several hidden layers to make predictions more accurate.

2.3 Attention Mechanism

The concept of the Attention mechanism came from the human visual attention mechanism (Butterworth and Cochran, 1980). When people perceive things visually, they usually do not observe the scene end-to-end. Instead, they tend to observe specific parts according to their needs. When people find that a scene has something they want to observe in a certain part, they will learn to pay attention to that part in the future when similar scenes appear. With RNN or LSTM, the information accumulation of several time steps is needed to connect long-distance interdependent features. However, the longer the distance is, the less likely it is to be captured effectively. In the Attention calculation process, the connection between any two words in a sentence is directly established through one calculation step. Thus, the distance between long-distance dependent features is greatly shortened, which is conducive to the effective use of these features. Obviously, it is easier to capture the long-distance interdependent features in sentences after the introduction of Attention. In figure 4, self attention can be described as mapping a query and a set of key-value pairs to an output. The calculation of Attention is mainly divided into three steps. The first step calculates the similarity between query and each key to get the weight. The second step uses a softmax function (Jean et al., 2015) to normalize these weights. Finally, the weight and the corresponding key value are weighted and summed to get the final Attention. Currently, in NLP research, the key and value are always the same, that is, key=value. In this part, we use self-attention, which is denoted as key=value=query (Firat et al., 2016).

$$\begin{aligned} &Attention(Q, K) = \\ &\sum_{n=1}^{n} Similarity(Q, K_i) * V_i \end{aligned} \quad (4)$$

2.4 MLP Layer

This layer is a fully connected layer that multiplies the results of the previous layer with a weight matrix and adds a bias vector. The ReLU (Jarrett et al., 2009) activation function is also applied in this layer. The final result vectors are finally input to the output layer.

2.5 Output Layer

This layer outputs the final classification result. It is a fully connected layer that uses softmax as an activation function.

3 Experiments and Evaluation

3.1 Data Preprocessing

The organizers of the competition provided the training data that included one question and a number of answers. Each of answer was to be classified into the categories: (Factual - TRUE, Factual - FALSE, Non-Factual). We extracted the questions and corresponding answers, and then concatenated them into the form of a question-answer pair. As all of the data was provided by the "Qatar Living" forum, the content primarily contained English text, and all non-english characters were ignored. We converted all letters into lower case to accommodate the known tokens in word2vec pretrained word vectors. We counted the sentence length of questions and answers. Most of them were no more than 80 words. Therefore, we set the length of the sentence to 80 words. The word2vec pretrained data was used to initialize the weight of the embedding layer. word2vec is a popular unsupervised machine learning algorithm to acquire word embedding vectors. We used 100-dimension word vectors to initialize the weight of the embedding layer.

3.2 Implementation

We used Keras with TensorFlow backend. The hyper-parameters were tuned in train and dev sets using the scikit-learn grid search function that can iterate through all possible parameter combinations to identify the one that provides the best performance. The optimal parameters found are as follows. The LSTM layer count is 2, and the dimension of the LSTM hidden layer (d) is 200. The dropout rate is 0.3. The training has a batch size of 128 and runs for 30 epochs. The results also revealed that the model using pre-trained word2vec vectors and an Adam optimizer achieved the best performance.

3.3 Evaluation Metrics

The system was scored based on Accuracy, macro-F1, and AvgRec where the "Factual - True" instances were considered to be positive, and the remaining instances to be negative.

3.4 Results and Discussion

To prove the advantages of our system architecture, we ran a 6-fold cross validation on different sets of layers. On training data, the trial data experiment results shown in Table 1:

Model	**F_1-score**
CNN	0.483
LSTM	0.498
BiLSTM	0.514
BiLSTM-Attention	**0.548**

Table 1: The trial data experiment results.

Our system achieved 0.548 accuracy on Subtask B. The evaluation results revealed that our proposed system showed considerable improvement over the average baseline, which we attribute to our LSTM with Attention architecture. Our system can effectively extract features from question and answer. Using this, prediction can be made on whether the answers are actually factual and whether the fact is true or not.

4 Conclusion

In this paper, we described our submission to the SemEval 2019 Workshop Task 8, which involved Fact-Checking in Community Forums. The proposed LSTM-Attention model combines LSTM and Attention. LSTM extracts local information within both the answer and question. The Attention Mechanism resolves the issue of poor learning effect on the long input sequence. The official results reveal that our system output performed all baseline algorithms and ranked 9th on Subtask B. In future work, we will query a search engine to fetch relevant documents from the Internet to achieve an improved classification system.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants No.61702443 and No.61762091, and in part by Educational Commission of Yunnan Province of China under Grant No.2017ZZX030. The authors would like to thank the anonymous reviewers and the area chairs for their constructive comments.

References

George Butterworth and Edward Cochran. 1980. Towards a mechanism of joint visual attention in human infancy. *International Journal of Behavioral Development*, 3(3):253–272.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism.

F. A. Gers, J Schmidhuber, and F Cummins. 2000. Learning to forget: continual prediction with lstm. *Neural Computation*, 12(10):2451–2471.

Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann Lecun. 2009. What is the best multi-stage architecture for object recognition? volume 12.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. *Computer Science*.

Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *Computer Science*.

Tsvetomila Mihaylova, Preslav Nakov, Lluis Marquez, Alberto Barron-Cedeno, and James Glass. 2018. Fact checking in community forums.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Computer Science*.

A C Tsoi and A D Back. 1994. Locally recurrent globally feedforward networks: a critical review of architectures. *IEEE Transactions on Neural Networks*, 5(2):229–39.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

DBMS-KU at SemEval-2019 Task 9: Exploring Machine Learning Approaches in Classifying Text as Suggestion or Non-Suggestion

Tirana Noor Fatyanosa[1], **Al Hafiz Akbar Maulana Siagian**[1,3], **Masayoshi Aritsugi**[2]
[1]Computer Science and Electrical Engineering
Graduate School of Science and Technology, Kumamoto University, Japan
[2]Big Data Science and Technology
Faculty of Advanced Science and Technology, Kumamoto University, Japan
[3]Indonesian Institute of Sciences, Indonesia
{fatyanosa,alha002}@dbms.cs.kumamoto-u.ac.jp
aritsugi@cs.kumamoto-u.ac.jp

Abstract

This paper describes the participation of DBMS-KU team in the SemEval 2019 Task 9, that is, suggestion mining from online reviews and forums. To deal with this task, we explore several machine learning approaches, i.e., Random Forest (RF), Logistic Regression (LR), Multinomial Naive Bayes (MNB), Linear Support Vector Classification (LSVC), Sublinear Support Vector Classification (SSVC), Convolutional Neural Network (CNN), and Variable Length Chromosome Genetic Algorithm-Naive Bayes (VLCGA-NB). Our system obtains reasonable results of F1-Score 0.47 and 0.37 on the evaluation data in Subtask A and Subtask B, respectively. In particular, our obtained results outperform the baseline in Subtask A. Interestingly, the results seem to show that our system could perform well in classifying Non-suggestion class.

1 Introduction

Nowadays, a huge number of texts are posted in online reviews or discussion forums. Such media can be a valuable source for obtaining a suggestion about products or services (Negi and Buitelaar, 2015; Negi et al., 2016). The obtained suggestion is not only useful for readers but also important information for stakeholders (Negi et al., 2016). Indeed, such advice can be used to improving the quality of products or giving helpful recommendations (Brun and Hagege, 2013). However, identifying a suggestion from a lot of reviews or comments needs extra effort and time. Moreover, such online texts are mostly in unstructured form (Negi et al., 2018; Negi and Buitelaar, 2017). Thus, automatically mining the suggestion from given texts is challenging and significant (Negi et al., 2016).

Suggestion mining is relatively a new research interest in text classification tasks (Negi and Buitelaar, 2015). Several studies have initiated to mining suggestions from online texts (Negi et al., 2018; Negi and Buitelaar, 2017; Negi et al., 2016; Negi, 2016; Negi and Buitelaar, 2015; Brun and Hagege, 2013; Ramanand et al., 2010; Dong et al., 2013; Wicaksono and Myaeng, 2012, 2013). Particularly, (Negi and Buitelaar, 2015; Brun and Hagege, 2013; Ramanand et al., 2010) have tried to identify suggestions from customer reviews. Meanwhile, (Negi, 2016; Dong et al., 2013; Wicaksono and Myaeng, 2012, 2013) have mined such advice by using Twitter or discussion forums dataset. Then, (Negi et al., 2018; Negi and Buitelaar, 2017) have utilized WikiHow and open domain corpora for their work. However, they concluded that it is not easy to identify suggestion texts automatically. In other words, it still has room to improving the classification result in the suggestion mining task. The task of suggestion mining from online reviews and forums, namely, Task 9 (Negi et al., 2019), is opened in the International Workshop on Semantic Evaluation 2019 (SemEval-2019).

This paper delineates the participation of DBMS-KU team in both Subtask A and Subtask B of Task 9 of SemEval-2019 (Negi et al., 2019). To address these two Subtasks, we utilize several approaches, namely, Random Forest (RF), Logistic Regression (LR), Multinomial Naive Bayes (MNB), Linear Support Vector Classification (LSVC), Sublinear Support Vector Classification (SSVC), Convolutional Neural Network (CNN), and Variable Length Chromosome Genetic Algorithm Naive Bayes (VLCGA-NB). The obtained results of our experiments are encouraging and show a promising improvement in identifying Suggestion and Non-suggestion.

The rest of this paper is organized as follows. Section 2 explains the problem definition, problem formulation, and dataset. Section 3 presents the tools and libraries used in this work. Section 4 describes our employed methods. Section 5 repre-

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 1185–1191
Minneapolis, Minnesota, USA, June 6–7, 2019. ©2019 Association for Computational Linguistics

sents our experiments that consist of data preprocessing, parameter, and evaluation measurement. Section 6 discusses our obtained results. Finally, we conclude this work in Section 7.

2 Problem Definition

Suggestion mining is a binary classification problem. Particularly, suggestion mining is a task that labels sentences as Suggestion or Non-suggestion. However, suggestion sentences can have very broad meaning. Thus, the domain and scope of the suggestion text classification should be described. The Task 9 of SemEval-2019 consists of Subtask A and Subtask B that are classifying a suggestion in intra-domain and cross-domain, respectively (Negi et al., 2019).

2.1 Problem Formulation

Suggestion text classification consists of assigning $suggestion, nonsuggestion$ to $(s_i, l_j) \in SxL$, where S is sentences and $L = [l_1, ..., l_n]$ is a set of *n* predefined labels. Each sentence is classified as Suggestion or Non-suggestion class.

2.2 Datasets

Dataset used in Task 9 of SemEval-2019 is divided into training, trial, and evaluation parts (Negi et al., 2019). The dataset consists of three columns: id, sentence, and label (see Table 1). The provided dataset is imbalanced in which, overall, Non-suggestion class is larger than Suggestion one.

3 Tools and Libraries

The common classification methods, such as Support Vector Machine, Random Forest Classifier, Linear Regression, and Naive Bayes application are facilitated by the most outstanding library for machine learning, namely, SciKit-Learn (Pedregosa et al., 2011). Correspond to its name, NLTK (Bird et al., 2009) is used as the toolkit for Natural Language Processing (NLP) operations such as tokenization, stemming, metrics, corpus, and classification. Pandas (McKinney, 2010) is chosen as the tools for collection and format the data because of its ease of use. The Keras library (Chollet et al., 2015) that runs on top of Tensorflow (Abadi et al., 2015) is also utilized for building high-level neural networks, i.e., for building the Convolutional Neural Network in this work. Furthermore, we utilize Seaborn[1] and Matplotlib library (Hunter, 2007) as confusion matrix visualization.

4 Classification Methods

This section details the classification methods used in our experiments on Suggestion classification.

4.1 Baseline

Baseline method provided by organizer utilizes the suggestion keyword, pattern string, and Part-Of-Speech (POS) Tagger matching. The list of suggestion keywords utilized in the baseline is "suggest", "recommend", "hopefully", "go for", "request", "it would be nice", "adding", "should come with", "should be able", "could come with", "i need", "we need", "needs", "would like to", "would love to", "allow", and "add". The baseline method also utilizes the wishes identification pattern string from (Goldberg et al., 2009). The POS tag of each word in the sentences is also done to collect Modal and Verb POS tag only. The classification is done by checking all words in the sentence. If the sentence contains one of the three matches, then the sentence is classified as a Suggestion class.

4.2 Common Classification Methods

Common classification methods, such as Support Vector Machine (SVM), Random Forest Classifier (RF), Linear Regression (LR), and Naive Bayes (NB) are employed in this research. Two types of SVM are utilized, that is, Linear Support Vector Machine Classifier (LSVC) and Sublinear Support Vector Machine Classifier (SSVC). The implementation of the common classification methods is available at (Fatyanosa, 2019c).

4.3 Variable Length Chromosome Genetic Algorithm-Naive Bayes

Variable Length Chromosome Genetic Algorithm - Naive Bayes (VLCGA-NB) is utilized for features selection. We follow the model and parameter from (Fatyanosa et al., 2018). The first step of VLCGA-NB is selecting initial features from keywords that appear in the Suggestion sentences but do not appear in the Non-suggestion ones in the training data. Within the randomly determined maximum chromosome size, these keywords are

[1]https://seaborn.pydata.org/generated/seaborn.heatmap.html

Table 1: Dataset example

ID	Sentence	Label
9636	Make ProgressRing control available for Windows Phone just like Win8.	1
9706	Either one has to use .NET to access a library or Microsoft advises to do a .NET app with native code in a WinRT component.	0
9709	Don't limit us artifically just because you don't like native developers.	0
9735	These page templates should be updated to use the medium-sized semibold font by default for the title text.	1
...	...	...

then randomly selected as genes in Genetic Algorithm (GA). Therefore, each chromosome within population will have different length with different genes. All populations resulting from the initialization then evolve through generation by passing the crossover, mutation, and selection operator. A number of children produced by crossover and mutation operator are based on Crossover Rate (CR) and Mutation Rate (MR). Two types of crossover are utilized in this research, viz., Union Crossover and Intersection Crossover. The mutation is done by changing the genes with another feature which is not contained in the chromosome. The gene which will be mutated within chromosome is selected by comparing the generated random value with the MR. If the random value is higher than MR, then the gene will be mutated. The purpose of the crossover operator is to help the algorithm to explore the search space, while the purpose of mutation operator is to exploit certain area in the search space. With these operators, there will be diversity within the population that can help to avoid early convergence. By ranking the fitness value using elitist selection, the next population for the next generation is selected from the prior population and the produced children. Only the chromosome with the highest ranking within the number of population will be selected. All these operators are then iterated until the maximum number of generations. The best chromosome produced in the last generation is then used as the Suggestion keywords in the baseline code provided by the organizer. GA, which is a well-known evolutionary algorithm, is one of the powerful stochastic and heuristic algorithms. The use of GA is legion as it can provide search space exploration through crossover operator and exploitation through mutation operator. Thus, GA is possible to search in a very wide search space and allow it to produce nearly optimal results. This ability becomes the motivation for feature selection using GA. However, the drawback of the GA is that it is not guaranteed to produce the global optimal, but instead satisfactory results. Moreover, GA requires parameter tuning to find the appropriate parameter based on the dataset and needs a longer runtime. Despite its drawback, we expect that GA can produce a limited number of Suggestion features which has a major contribution to the Suggestion classification in this work. The implementation of the VLCGA-NB is available at (Fatyanosa, 2019b).

4.4 CNN

For our purpose in this work, we follow the Keras model's architecture from (Chollet, 2017) as shown in Figure 1. This architecture tends to obtain high accuracy when applied on the Newsgroup dataset. The text classification using CNN is done in four steps. First, all sentences are converted into word index order. Only 20,000 frequently words with the upper limit length of 1000 words will be considered. Next, 100-dimensional Global Vectors for Word Representation (GloVe) embeddings are utilized as the embedding matrix. Then, this matrix is loaded into Embedding layer of Keras. Finally, the Softmax function is used in the final layer of CNN. The implementation of the CNN is available at (Fatyanosa, 2019a). Although there are several pre-trained word vectors, GloVe and word2vec are considered as the most popular vectors (Lee et al., 2016). Based on (Pennington et al., 2014), their GloVe vector has outperformed other word representations in terms of word comparison, correlation, and named entity recognition. We thus use the GloVe vector as our pre-trained word embedding in this work.

5 Experiments

We conducted the experiments with the seven classification methods in this section. We employed the datasets from both Subtasks for evaluating the

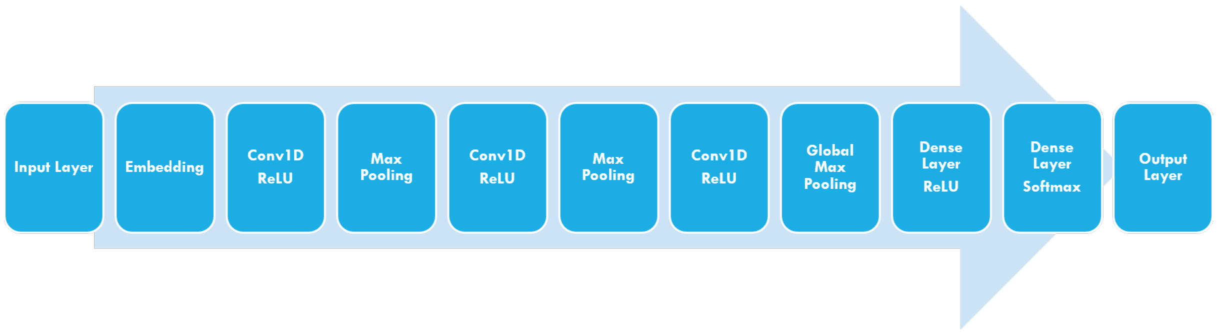


Figure 1: CNN Architecture from (Chollet, 2017)

performance of those classification methods. We compared the performance of seven classification methods against the baseline provided by the organizer.

5.1 Data Preprocessing

We performed a sequence of preprocessing steps to address noise in the data. For RF, MNB, LSVM, and LR, all features were converted into numerical feature vectors using the Term Frequency-Inverse Document Frequency (tf-idf) from *SciKit-Learn* with parameter *sublinear_tf=True, min_df=5, norm='l2', encoding='latin-1', ngram_range=(1, 2), stop_words='english'*. The obtained number of features was 3166. While the parameters of SSVM were *sublinear_tf=True, analyzer='word', tokenizer=tokenize, lowercase=True, ngram_range=(1, 1), stop_words=en_stopwords, norm='l2', min_df=3*. The obtained number of features was 3844.

The series of VLGCA-NB preprocessing was different from other classification methods because it did not need the vector form of words. All words were converted to lowercase. Number, stop words, punctuation, non-English words, non-alphabetic characters, and words smaller than two characters were removed, lemmatization was performed, and all contractions were replaced by their real words or phrases. The number of features was decreased to 493 after the preprocessing step.

5.2 Parameters

To apply CNN to the classification, we used fixed parameters with $maximumEpoch = 10$ and $batchSize = 100$. We also employed fixed parameters for VLCGA-NB with $Populationsize = 100$, $Generationsize = 50$, $Crossoverrate = 0.7$, and $Mutationrate = 0.3$.

RF, CNN, and VLCGA-NB are stochastic algorithms which mean the results will differ for each run. Therefore, those algorithms were run five times. The best result among the five attempts was selected for comparison with other algorithms.

5.3 Evaluation Measurement

Classifier performance evaluation using accuracy is often considered as a suited measurement. However, the datasets from both subtasks were imbalanced. Majority class is often reckoned by the classifier, thus, higher accuracy will be achieved for it. Therefore, in this research, we used Precision, Recall, and F1-Score as the main evaluation measures. Accuracy measurement (Equation (1)) was still used in the fitness function of VLCGA-NB. F1-Score (Equation (4)) computation relied on Precision (Equation (2)) and Recall (Equation (3)) measurements. As the Suggestion results were more concerned, the evaluation of this competition was the Suggestion results' F1-Score.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$F1\text{-}Score = 2x\frac{Precision x Recall}{Precision + Recall} \quad (4)$$

$where:$
$TP =$ True Positive
$TN =$ True Negative
$FP =$ False Positive
$FN =$ False Negative

Evaluation measurement for VLCGA-NB was done in every generation using Fitness Function based on the result of Naive Bayes classification.

The Fitness value was found by addition of accuracy, F1-Score of Suggestion and F1-Score of Non-suggestion, which were defined as follows:

$$\begin{aligned} Fitness = Accuracy + F1\text{-}Score_{suggestion} \\ + F1\text{-}Score_{non-suggestion} \end{aligned} \quad (5)$$

6 Results and Discussion

In this section, we evaluated the classification performance of the seven classification methods with the baseline for both Subtasks. The methods performance was evaluated on Precision, Recall, and F1-Score metrics, except for the VLCGA-NB, we still used accuracy in the fitness function.

A typical observation from the confusion matrix produced in this research was that the number of correct classification was higher than the number of misclassified for the Non-suggestion class, except for baseline of Subtask A. This result was unvaried across all methods, with a little difference of the whole classification count. Though Subtask A aimed to classify Suggestion in the same domain, the number of correct classification of Suggestion class was lower than the number of misclassified for most of the classification methods. In particular, baseline and SSVC obtained better results than other utilized methods. Furthermore, as Subtask B aimed to classify Suggestion in the different domain, eventually it was hard for all classification methods to obtain even fair results. All of them failed to obtain a higher number of correct classification of the Suggestion class.

Table 2 shows the precision, recall, and F1-Score comparisons for each class in Subtask A. We noted that the obtained result of all classification methods outperformed that of the baseline for the Non-suggestion class. MNB yielded the best results with 0.95.

In terms of F1-Score of the Suggestion class, refer to Table 2, we noted that RF, SSVC, and VLCGA-NB obtained a competitive result outperforming baseline for Suggestion class. The highest F1-Score was obtained by SSVC at 0.47. RF and VLCGA-NB produced F1-Score at 0.29 and 0.31, respectively. Overall, note that MNB and SSVC obtained the best F1-Score for Non-suggestion and Suggestion classes, respectively.

Table 3 shows the experimental results for the dataset in Subtask B. We noted that the F1-Score of Non-suggestion yielded a good result with the higher F1-Score obtained by RF classifier. However, the F1-Score of Suggestion class produced poor finding, except for the baseline. F1-Score of Suggestion class using the baseline achieved surprisingly well considering their simplicity, which yielded 0.73. A possible reason for this finding might be that the manually selected keywords and patterns based on which usually used to suggest something would make use of the common Suggestion sentence that a machine might not be able to discover. A possible problem with the baseline approach was probably that the keywords and patterns for Suggestion class might be also used for Non-suggestion class. Therefore, it might be difficult for baseline to define which keywords and patterns actually used in Suggestion sentences. This could be proven from the F1-Score results for the Non-suggestion class in Subtask A which yielded the lowest result of 0.59.

Regarding the number of features selected by VLCGA-NB, the features were decreased from 493 to 372. Refer to our defined expectation in 4.3, VLCGA-NB was able to produce a limited number of features which has major contribution to the Suggestion classification. This could be proven from its F1-Score results which yielded higher value compared to the baseline in Subtask A.

7 Conclusion

This paper has described our approach for participating in both Subtask A and Subtask B of Task 9 of SemEval-2019, that is, suggestion mining from online reviews and forum (Negi et al., 2019). Our approach explored and compared various classification methods, namely, Random Forest, Logistic Regression, Multinomial NB, Linear SVC, Sublinear SVC, CNN, and VLCGA-NB. Since the datasets provided by the organizer were imbalanced data, it was more important to correctly classify a sentence as a Suggestion class. Thus, the F1-Score of the Suggestion class was more considered. Compared to the baseline, all algorithms performed better classification for the Non-suggestion class in both Subtask A and Subtask B. In contrast, they performed worse classification than the baseline for the Suggestion class in both Subtask A and Subtask B. This poor performance was except for the RF, SSVC, and VLCGA-NB that could outperform the baseline for classifying the Suggestion class in Subtask A. Based on our results, we observed that besides the imbalanced data, the implicit meaning problem related to the

Table 2: Metrics Report Subtask A

Method	**Precision**		**Recall**		**F1-Score**	
	Non suggestion	**Suggestion**	**Non suggestion**	**Suggestion**	**Non suggestion**	**Suggestion**
Baseline	0.98	0.16	0.42	0.92	0.59	0.27
Random Forest	0.92	0.34	0.94	0.25	0.93	0.29
Logistic Regression	0.9	0.36	0.98	0.1	0.94	0.16
Linear SVC	0.91	0.21	0.87	0.3	0.89	0.25
Sublinear SVC	0.97	0.35	0.84	0.75	0.9	0.47
Multinomial NB	0.9	0.62	1	0.06	0.95	0.11
CNN	0.9	0.08	0.98	0.01	0.94	0.02
VLCGA-NB	0.92	0.45	0.97	0.23	0.94	0.31

Table 3: Metrics Report Subtask B

Method	**Precision**		**Recall**		**F1-Score**	
	Non suggestion	**Suggestion**	**Non suggestion**	**Suggestion**	**Non suggestion**	**Suggestion**
Baseline	0.82	0.69	0.74	0.78	0.78	0.73
Random Forest	0.59	0.49	0.88	0.15	0.93	0.29
Logistic Regression	0.58	0.41	0.95	0.05	0.72	0.08
Linear SVC	0.6	0.55	0.9	0.17	0.72	0.26
Sublinear SVC	0.64	0.68	0.9	0.29	0.74	0.37
Multinomial NB	0.57	0.13	0.97	0.01	0.72	0.01
CNN	0.58	0.51	0.96	0.05	0.73	0.1
VLCGA-NB	0.6	0.69	0.96	0.11	0.74	0.19

Suggestion class was also the challenge of the suggestion mining. The feature selection corresponds with the Suggestion class will be our future intention. In addition, it might be valuable to inspect further the use of our approach to other text classification tasks such as deceptive opinions (Siagian and Aritsugi, 2017, 2018) and fake news identifications.

Acknowledgments

The authors acknowledge the anonymous reviewers for their valuable comments and suggestions regarding this paper. A. H. A. M. Siagian thanks the scholarship support from RISET-Pro (Research and Innovation in Science and Technology Project) KEMENRISTEKDIKTI (Ministry of Research, Technology and Higher Education of the Republic of Indonesia).

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Caroline Brun and Caroline Hagege. 2013. Suggestion mining: Detecting suggestions for improvement in users comments. volume 70, pages 199–209. Research in Computing Science.

François Chollet et al. 2015. Keras. `https://keras.io`.

François Chollet. 2017. Using pre-trained word embeddings in a Keras model. In *The Keras Blog*. `https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html`.

Li Dong, Furu Wei, Yajuan Duan, Xiaohua Liu, Ming Zhou, and Ke Xu. 2013. The automated acquisition of suggestions from tweets. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, AAAI'13, pages 239–245. AAAI Press.

Tirana Noor Fatyanosa. 2019a. Neural network for text classification. `https://github.com/TiraNosa/NeuralNetworkForTextClassification`.

Tirana Noor Fatyanosa. 2019b. Text classification using vlcga-nb. https://github.com/TiraNosa/Text-Classification-using-VLCGA-NB.

Tirana Noor Fatyanosa. 2019c. Text classification with scikit-learn. https://github.com/TiraNosa/Text-Classification-with-Scikit-Learn.

Tirana Noor Fatyanosa, Fitra A. Bachtiar, and Mahendra Data. 2018. Feature Selection using Variable Length Chromosome Genetic Algorithm for Sentiment Analysis. In *2018 International Conference on Sustainable Information Engineering and Technology (SIET)*, Malang.

Andrew B. Goldberg, Nathanael Fillmore, David Andrzejewski, Zhiting Xu, Bryan Gibson, and Xiaojin Zhu. 2009. May All Your Wishes Come True: A Study of Wishes and How to Recognize Them. In *The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 263–271.

J. D. Hunter. 2007. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95.

Yang-Yin Lee, Hao Ke, Hen-Hsen Huang, and Hsin-Hsi Chen. 2016. Combining word embedding and lexical database for semantic relatedness measurement. In *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW '16 Companion, pages 73–74, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Wes McKinney. 2010. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, pages 51 – 56.

Sapna Negi. 2016. Suggestion mining from opinionated text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics – Student Research Workshop*, pages 119–125.

Sapna Negi, Kartik Asooja, Shubham Mehrotra, and Paul Buitelaar. 2016. A study of suggestions in opinionated texts and their automatic detection. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 170–178.

Sapna Negi and Paul Buitelaar. 2015. Towards the extraction of customer-to-customer suggestions from reviews. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2159–2167.

Sapna Negi and Paul Buitelaar. 2017. Inducing distant supervision in suggestion mining through part-of-speech embedding. arXiv:1709.07403. Version 2.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. SemEval-2019 Task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Sapna Negi, Maarten de Rijke, and Paul Buitelaar. 2018. Open domain suggestion mining: Problem definition and datasets. arXiv:1806.02179. Version 2.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Janardhanan Ramanand, Krishna Bhavsar, and Niranjan Pedanekar. 2010. Wishful thinking: Finding suggestions and 'buy' wishes from product reviews. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, CAAGET '10, pages 54–61, Stroudsburg, PA, USA. Association for Computational Linguistics.

Al Hafiz Akbar Maulana Siagian and Masayoshi Aritsugi. 2017. Combining word and character n-grams for detecting deceptive opinions. In *2017 IEEE 41st Annual Computer Software and Applications Conference*, COMPSAC, pages 828–833, Washington, DC, USA. IEEE Computer Society.

Al Hafiz Akbar Maulana Siagian and Masayoshi Aritsugi. 2018. Exploiting function words feature in classifying deceptive and truthful reviews. In *2018 Thirteenth International Conference on Digital Information Management*, ICDIM, pages 51–56, Washington, DC, USA. IEEE Computer Society.

Alfan Farizki Wicaksono and Sung-Hyon Myaeng. 2012. Mining advices from weblogs. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 2347–2350, New York, NY, USA. ACM.

Alfan Farizki Wicaksono and Sung-Hyon Myaeng. 2013. Automatic extraction of advice-revealing sentences foradvice mining from online forums. In *Proceedings of the Seventh International Conference on Knowledge Capture*, K-CAP '13, pages 97–104, New York, NY, USA. ACM.

DS at SemEval-2019 Task 9: From Suggestion Mining with neural networks to adversarial cross-domain classification

Tobias Cabanski
t.cabanski@posteo.de

Abstract

Suggestion Mining is the task of classifying sentences into suggestions or non-suggestions. SemEval-2019 Task 9 sets the task to mine suggestions from online texts. For each of the two subtasks, the classification has to be applied on a different domain. Subtask A addresses the domain of posts in suggestion online forums and comes with a set of training examples, that is used for supervised training. A combination of LSTM and CNN networks is constructed to create a model which uses BERT word embeddings as input features. For subtask B, the domain of hotel reviews is regarded. In contrast to subtask A, no labeled data for supervised training is provided, so that additional unlabeled data is taken to apply a cross-domain classification. This is done by using adversarial training of the three model parts label classifier, domain classifier and the shared feature representation. For subtask A, the developed model archives a F1-score of 0.7273, which is in the top ten of the leader board. The F1-score for subtask B is 0.8187 and is ranked in the top five of the submissions for that task.

1 Introduction

For getting feedback from costumers or users, an organization often uses forums and social media channels. Also ratings of products on rating platforms can be an useful feedback to make a product better. The feedback from a customer can be in the form of a suggestion which appears in a rating text or is directly asked from the customer. The task of suggestion mining can be defined as the extraction of sentences that contain suggestions from unstructured text (Negi et al., 2018). SemEval-2019 Task 9 Subtask A provides the challenge to do suggestion mining on data from an online suggestion forum. For that subtask, a train and validation set is provided so that it is possible to apply supervised training. For subtask B, suggestions in hotel reviews should be identified. An additional difficulty for that subtask is that no labeled data is given except a small validation set, which is not allowed to be used for supervised training. For both tasks, silver standard datasets are allowed to use, which means that data that is likely to belong to a certain class can be taken as long as it is not manually labeled. A more detailed task description can be found in (Negi et al., 2019).

2 Data

The dataset for subtask A provides an overall count of 8500 examples, where 6415 examples are labeled as non-suggestion and 2085 as suggestion. Also a trial dataset is provided that contains 592 examples, divided in 296 examples for each class. Every example contains only one sentence, which could be part of a whole post in the forum where it was extracted.

The domain of software suggestion forum posts in general provides more balanced data than other domains, for example hotel reviews. Also the domain contains very specific vocabulary which is frequently used in software development, so that it can be difficult to use a trained model of this domain for other domains (Negi et al., 2018).

For subtask B, only a validation dataset is provided. The set contains an overall count of 808 examples with 404 examples for each class. As mentioned in the introduction, it is not allowed to use the validation data for supervised learning for this subtask. The data is only allowed to be used for model evaluation and error analysis and also for automatic hyperparameter tuning. The presented solution in this paper uses the validation data for early stopping at a fixed count of train steps after the best score is reached. The model state at the best score is then returned and used for the prediction of the test data.

For both subtasks, additional data is allowed that is not manually labeled. In this work, the hotel review dataset, which is presented in (Wachsmuth et al., 2014), is used to apply cross-domain classification for subtask B. The dataset comes with nearly 200k examples of hotel reviews without labels.

3 Related Work

The task of text classification improved a lot during the last years. In the past, machine learning techniques like support vector machines were used to assign a class to a text. In (Joachims, 1998), a text classification with support vector machines is presented. For that, a document vector is extracted for the whole text and used as the feature vector for the classification.

Since such methods can provide a good base line today, the increasing popularity of neural network approaches provides new methods that can classify texts more exactly. Especially the introduction of word embeddings in (Mikolov et al., 2013) was a big step forward in the field of text processing and opened new opportunities for many natural language processing tasks. Also for the task of text classification, word embeddings are useful features and can lead to good results. Since the release of these word embeddings, many other word embedding approaches have been introduced. A very recent one is shown in (Devlin et al., 2018) and is called BERT: Bidirectional Encoder Representations from Transformers. These embeddings are the result of the training process of transformer, which is described in (Vaswani et al., 2017) and delivers a state-of-the-art method for different natural language generation tasks, especially for translation.

To use the word embeddings as features for text classification, a commonly used approach is Long-short term memory (LSTM), which is described in (Hochreiter and Schmidhuber, 1997). The advantage of using LSTM cells over support vector machines as classifier is the processing of features in time steps. By passing a single word embedding into a single time step of the LSTM, every feature is processed separately. Since the features are processed one after the other, also the order of the features has influence on the classification process. In addition to that, LSTM cells have a state that enables them to save information for many previous time steps. For text classification, this can be useful when there are connections between words in a text that are far apart.

Another method to process word embeddings are convolutional neural networks (CNN), which are introduced in (LeCun and Bengio, 1998). With the ability to extract features of two-dimensional input data by defining a sliding window of variables, the method is often used for image processing. But also good results for text classification are reported, for example in (Kim, 2014). The results show that even a simple CNN with one layer performs very well and a tuning of hyperparameters brings an improvement of the performance.

For subtask B, the focus gets in the direction of methods for cross-domain classification. By the introduction of Generative Adversarial Nets (GAN) in (Goodfellow et al., 2014), a new way for training neural networks was provided that leads to new opportunities for different task, especially image generation. In (Chen and Cardie, 2018) is shown that this training technique could also be used for cross-domain classification of texts of different domains. As the main four components, a shared feature extractor, a domain feature extractor, a text classifier and a domain discriminator are introduced. The main goal of that system is to learn a domain invariant feature representation by training the shared feature extractor with the discriminator and the text classifier. The discriminator learns to separate the domains and the training goal for the shared feature extractor is to increase the loss of the discriminator. The extracted features become invariant for the domains, so that the text classifier results improve for the domain where no labels are given.

4 Models

In this section, the models for subtask A and B are presented. The overall idea of the model for subtask A is using an ensemble of LSTM and CNN networks. As input features, pre-trained BERT embeddings for the texts are used.

For subtask B, the idea is to extend the model from subtask A with a domain discriminator and shared features. Since that adds a lot of parameters to the model, the text classifier has a simpler structure than in subtask A and uses only CNNs for classification. The full TensorFlow implementation of the models can be found at GitHub.[1]

[1]https://github.com/tocab/semeval2019Task9

4.1 BERT embeddings

For both subtasks, BERT embeddings are used to create a representation of the text. The TensorFlow implementation, which is openly available and comes with pre-trained multi-language embeddings, is taken for that.[2] The model for creating the embeddings is the small uncased model, which has been trained on lower cased wikipedia texts. It has a total count of twelve layers and a layer size of 768 in each hidden layer. The whole model has a 110 million parameters in total.

To extract the embeddings out of the model, the text gets tokenized and mapped into a sequence of integers by using the vocabulary of the pre-trained model. This representation is then given into the network, where it passes the different transformer layers. The embeddings are delivered by the hidden layers of the model. In this project, the output from the last four layers before the output layer is taken as the representation of a word, so that every word is represented by a vector of the shape $(4, 768)$.

4.2 Subtask A

For subtask A, a text classification ensemble of LSTM and CNN is built. To bring all sentences to the same length, a maximum sequence length of 40 is defined. With that sequence length, for around 95% of all train data sentences all words are taken as input. Only for the remaining 5% which have more than 40 words, the texts are cut to the maximum sequence length. If a text is shorter than 40 words, it is filled with zeros. Using the batch size of 64, the input shape for a batch for the training process is $(64, 40, 768)$ for every of the four extracted input features from BERT.

One problem of the data is the imbalance of the classes. When taking random batches out of the whole dataset, it is likely that the count of one class is always higher than of the other. The algorithm learns to predict the class with the higher example count with a higher probability. To avoid that, the technique of oversampling is applied to the training process. The data is separated into two sets, each for every of the two classes and then fed into the network alternately. When all examples of the class with the lower count were used as training input, the set gets repeated so that these examples occur more often as training input.

[2]https://github.com/google-research/bert

The model structure for subtask A can be divided into the three following main parts:

- Processing of single words with dense layers.
- Processing of the whole text with LSTM cells.
- Processing of sliding windows through the text with CNN.

For the processing of the dense layer and the LSTM, separated graphs are created for each of the input features from BERT. As mentioned before, four embeddings for a word are gathered, each of a different hidden layer of the transformer. Thus four graphs of dense layers and LSTM layers are created, each for processing a different embedding type of the input text.

The first step is a transformation of every single input word with a dense layer. This approach is applied to focus on single signal words that can occur in the text. For example, the occurrence of the word *recommend* could be a hint for a suggestion, without regarding other words. The outputs of the single word processings are concatenated and forwarded to a dense layer to reduce the dimension.

The LSTM is represented by two cells to realize a bidirectional approach. The output of the two cells is concatenated and followed by a `GlobalMaxPool`-Layer, which takes the maximum of the output's timestep axis to bring it into a one-dimensional representation. To do a further feature transformation, a dense layer is applied to the output. The result is concatenated to the output of the previous described single word approach.

Unlike the single word processing dense layer and the LSTM, the CNN approach processes all four BERT features for the words in the same network. When using CNNs for image processing, the colors of an image are arranged as additional channels that the CNN can process. For feeding the CNN with all BERT features for the words, they are shaped similarly to an image and can be seen as the channels of the word. By using that approach, the words are given with four channels into the CNN. The output of the CNN is then processed by a dense layer. The CNN approach can be seen as an bag of words approach, which takes the words within a sliding window until the end of text is reached. The amount of words is fixed, the approach is build for each of 2-5 words.

At the end of the processing, the dense- and LSTM-features and the CNN-features are concatenated and given to the classification layer, which is composed of two dense layers. For more robust predictions, three graphs are build to get three predictions, the final result is formed by the mean. For training the model, the cross-entropy-loss is used. The model gets optimized with `AdamOptimizer`. On every training step, the model is validated with the provided validation dataset, and the model weights on the best F1-score are taken to predict the test examples.

4.3 Subtask B

For subtask B, a similar model as in (Chen and Cardie, 2018) is built to do cross-domain classification. The model in this work is composed of three major parts:

- **Label classifier**: Model that predicts if an example is a suggestion.
- **Domain classifier**: Model for the prediction of the domain of an example.
- **Shared features**: Model that applies a transformation on the input features.

The training of the model can be split into two phases: The pre-training phase of the supervised label classifier and the adversarial training of the domain classifier and the shared features.

In the pre-training phase, the model uses supervised training like for subtask A. In this phase, the label classifier and the shared features are trained to get the best score on the suggestion data of the domain of online suggestion forums. The shared features get the word embeddings as input and apply a CNN on each BERT embedding. The size of the CNNs is the same as the length of the embedding, so that the projected word features are of the same shape as the input features.

In the next step, the projected features are classified with the label classifier. Unlike to subtask A, only the CNN part is used for the classification because of the amount of additional parameters of the shared features and the domain classifier. The CNN works like in subtask A and processes the four BERT features as single channels. In this task, a sliding window of the word counts from 2-6 words is taken. The optimization is applied with `AdamOptimizer` and stops when the best score is reached on the validation data of subtask A.

	train	**val**	**test**	**pred**
#examples	8500	592	833	833
#suggestions	2085	296	87	133
#non-suggestions	6415	296	746	700

Table 1: Count of the examples for the different datasets and the prediction on the test set for subtask A.

In phase two, the domain classifier starts with the training and learns to choose the right domain for the examples. To do that, also examples of the unlabeled hotel review dataset are given into the net. The domain classifier has the same structure as the text classifier and uses CNNs to find the right label for an example.

After one train step of the domain classifier, the shared features are retrained in an adversarial way to maximize the loss of the domain classifier. To realize this, the parameters are trained with the switched labels for the hotel review examples which are marked as suggestions for this training step.

After the training step of the domain classifier and the shared features, the validation examples of subtask B are used to predict a score. To do that, the examples are predicted with the text classifier, which uses the updated shared features to make a prediction if the example is a suggestion. Early stopping is used to find the best model with the validation data, so the model stops at the maximum F1-score for the validation set for subtask B.

5 Results

In this section, the results for subtask A and B are discussed. Also the test data, which has been provided to the participants after the evaluation phase, is used for the analysis.

5.1 Subtask A

For subtask A, the best model reached a F1-score of 0.7273 for the test data. This model archived a validation F1-score of 0.875 which is a noticeable difference to the test data score. To find the difference in the datasets, the example counts are compared in Table 1. It can be seen that there are differences in the ratio of the data. While the train data contains about 25% of examples for suggestions, for the test data there are only about 15%. Since oversampling is used to tackle the problem of class imbalance, this difference of train and test

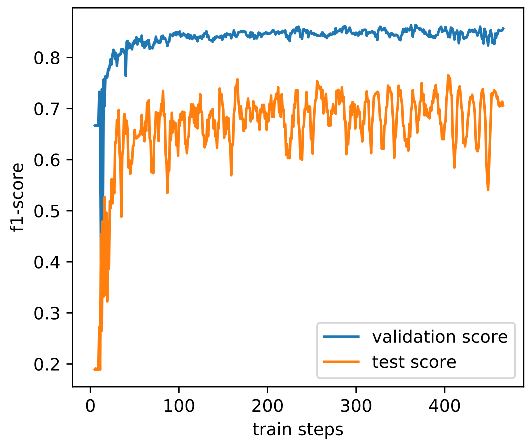


Figure 1: Comparison of the F1-scores for validation and test set during training for subtask A.

data should not have much influence on the result. Also the counts of predicted classes for the test set can be seen in the table, which show a similar amount like the test classes, but has some more predictions for the class of suggestions.

Another factor for the gap between validation and test score could be explained by very different examples in the two sets. In addition to that, the validation set may be better represented by the train set. That could lead to bad results for the test data when stopping at a good F1-score for the validation data. For analyzing this, another train run is started and the curve for the F1-score for the validation set and the test set plotted. The outcome can be seen in Figure 1.

It can be seen that the test F1-score is constantly lower than the validation score. Also there are much variations in the test score, even in a late train phase. Overall it can be determined that the train data describes the validation data better than the test data with the given model for subtask A.

5.2 Subtask B

The model for subtask B reached a final F1-score on the test data of 0.8187. In comparison to subtask A, it can be seen that the final score on the test data is higher, although no labeled data is given for that task except the validation data. The reason for this could be the use of the external hotel review dataset that inputs many new examples into the model. The overall count of examples for hotel reviews is much higher than the labeled data in subtask A, so that the higher score can be explained with the presence of more data.

	val	**test**	**pred**
#examples	808	824	824
#suggestions	404	348	402
#non-suggestions	404	476	422

Table 2: Subtask B dataset and prediction counts.

Also the validation score of 0.884 doesn't differ to the test score as much as in subtask A. This also shows that the use of external data improves the overall result of the model. The validation data for subtask B was used to apply early stopping and saving the weights at the best validation F1-score. Like for subtask A, the class counts for the different datasets are shown in Table 2. It can be seen that the distribution of the classes in the test and validation set is more equal than in subtask A, what could be a reason why the model archived better results. To verify this, another training run is started to compare the test and validation F1-score over the training epochs. The results can be seen in Figure 2. The validation data score for subtask A and B and the test score for subtask B is plotted for every training step. Since the training is separated into two phases, the left graph shows the scores for the pre-training phase and the right graph for the adversarial training.

In the pre-training phase of the model, the score of the validation data of subtask A improves as expected since supervised training is performed. Also it can be seen that the subtask B data score shows a high variance over the epochs, but decreases slightly. Over all epochs, the test and validation score of subtask B is nearly the same what could be a hint that the difference between the datasets is very small. This is confirmed in phase two of the training, where the validation and test score increase in the first epochs. Although the validation score reaches a higher peak for the F1-score, the peak of the test score is found in around the same region of train steps. The validation score for subtask A decreases in the adversarial train phase, what could be caused by a overfitting to the hotel review domain. Also the validation score of subtask B decreases after about 50 epochs. The reason for that could be that the amount of non-suggestions in very high in the hotel reviews data, so that the model unlearns to distinguish between the two classes.

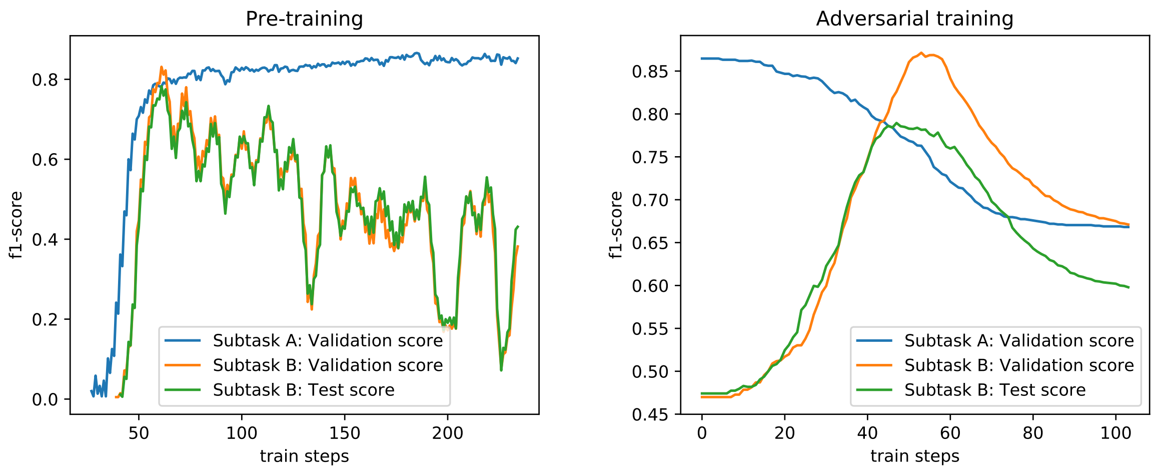


Figure 2: Two training phases for subtask B: First the pre-training with the subtask A data, then the adversarial training with the hotel review dataset.

6 Conclusion and Future Work

In this work, for each subtask of SemEval-2019 Task 9 a solution is presented. For subtask A, a supervised model is built on the neural network techniques CNN and LSTM. As input features, BERT word embeddings are taken, which are pre-trained on huge datasets. One problem in subtask A is the class imbalance of the data, which is tackled with oversampling. Another problem that occurred during the evaluation of the training phase is the difference of examples in the validation and the test set, what could be one of the reasons why the validation score is much higher than the test score. In future works, it can be tried to extend the labeled data with additional unlabeled data to tackle the problem of class imbalance and too few examples. Since extending the data works for subtask B, it could also work for subtask A and the domain of suggestion forum posts.

Subtask B sets the task to classify sentence as suggestion or non-suggestion for the domain of hotel reviews. Unlike to subtask A, only a small validation set is given as labeled data which is not allowed to be used for supervised training. To solve the problem of not having labeled data, the technique of cross-domain classification has been used. This is done by building a neural network model, which is trained in an adversarial way. Like in subtask A, a classification model for the suggestions is given. In addition to that, a shared feature representation and a domain classifier are added. The domain classifier is trained to assign the right domain label to a sentence. The shared feature representation is trained adversarial to the domain classifier, so that it learns to generate a global representation for both domains. For that, an unlabeled external dataset is taken which contains examples for the domain of hotel reviews.

The results for subtask B show that the adversarial training can improve the F1-score for the domain with no labeled data. This happens with the cost of lowering the score for the labeled data, on which the model was pre-trained. Also the score for the hotel review data falls down after the peak is reached. This makes it necessary to have at least a small dataset which contains labeled data for subtask B to measure the score during the training and stop when best score has been reached. For future work, it could be tried to develop a better shared features method where a good feature representation for both domains is formed. That would give a classifier that could be used to predict sentences of both domains. Another improvement could be archived by developing a model where the curve for the unlabeled data doesn't fall down that sharply in the late train phase. This could lead to a method were no labeled data is needed to stop the training at a good point.

References

Xilun Chen and Claire Cardie. 2018. Multinomial adversarial networks for multi-domain text classification. *CoRR*, abs/1802.05694.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML'98, pages 137–142. Springer-Verlag.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Yann LeCun and Yoshua Bengio. 1998. The handbook of brain theory and neural networks. chapter Convolutional Networks for Images, Speech, and Time Series, pages 255–258. MIT Press.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Sapna Negi, Maarten de Rijke, and Paul Buitelaar. 2018. Open domain suggestion mining: Problem definition and datasets. *CoRR*, abs/1806.02179.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, Gregor Engels, and Tsvetomira Palakarska. 2014. A review corpus for argumentation analysis. In *Proceedings of the 15th International Conference on Intelligent Text Processing and Computational Linguistics*, pages 115–127, Berlin Heidelberg New York. Springer.

Hybrid RNN at SemEval-2019 Task 9: Blending Information Sources for Domain-Independent Suggestion Mining

Aysu Ezen-Can
SAS Inst.
aysu.e.can@gmail.com

Ethem F. Can
SAS Inst.
ethfcan@gmail.com

Abstract

Social media has an increasing amount of information that both customers and companies can benefit from. These social media posts can include Tweets or be in the form of vocalization of complements and complaints (e.g., reviews) of a product or service. Researchers have been actively mining this invaluable information source to automatically generate insights. Mining sentiments of customer reviews is an example that has gained momentum due to its potential to gather information that customers are not happy about. Instead of reading millions of reviews, companies prefer sentiment analysis to obtain feedback and to improve their products or services.

In this work, we aim to identify information that companies can act on, or other customers can utilize for making their own experience better. This is different from identifying if reviews of a product or service is negative, positive, or neutral. To that end, we classify sentences of a given review as **suggestion** or **not suggestion** so that readers of the reviews do not have to go through thousands of reviews but instead can focus on actionable items and applicable suggestions. To identify suggestions within reviews, we employ a hybrid approach that utilizes a recurrent neural network (RNN) along with rule-based features to build a domain-independent suggestion mining model. In this way, a model trained on electronics reviews is used to extract suggestions from hotel reviews.

1 Introduction

With the growth of social media usage, the interest in text mining approaches has increased. One task that has gained momentum recently is sentiment analysis where the goal is to determine opinions/emotions from a text input, generally a product or service review. Different approaches have been proposed for sentiment analysis task such as multilingual models to be used with limited data (Can et al., 2018) and sentiment lexicons (Banea et al., 2008). Twitter posts also has been one source of reviews to be mined in terms of sentiment (Pak and Paroubek, 2010; Ezen-Can and Can, 2018; Tellez et al., 2017).

The task of suggestion mining is similar to sentiment analysis in that the input is the same (e.g., customer reviews). However, the output of a sentiment analysis model and a suggestion mining model is different. While sentiment classifiers focus on grouping reviews as positive or negative, suggestion mining models identify the reviews that have suggestions/actionable items/advice to other people/service providers.

In this paper, we present a suggestion mining model that takes product/service reviews and classifies each sentence in a given review as suggestion or not suggestion. To that end, we employ a hybrid LSTM model that utilizes both the textual reviews and features extracted from a rule-based approach.

2 Related Work

For the suggestion mining task, there is not a large body of work in the NLP community. (Brun and Hagege, 2013) use a corpus of reviews of printers made by different manufacturers. Their approach relies on linguistic information such as thesaurus, parser and patterns. (Goldberg et al., 2009) address the task of suggestion mining as a 'wish detection' task and use templates to detect wishes on product reviews and political discussion posts. (Dong et al., 2013) focus on Tweets and classify them as containing suggestion or not by using factorization machines. (Wicaksono and Myaeng, 2013) employed Hidden Markov models with three different sets of features: syntactic, contextual, and sentence informativeness features.

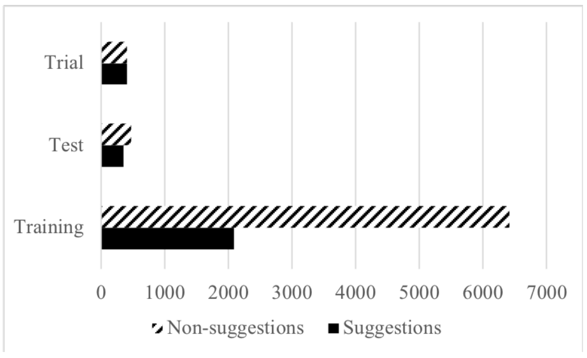


Figure 1: Distribution of classes in the training, trial and test sets.

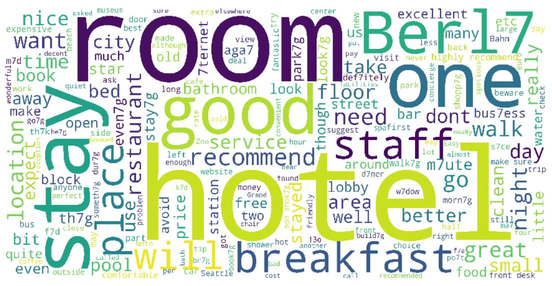


Figure 2: Word cloud of the reviews in the test set.

Recently, (Negi and Buitelaar, 2017) collected a new corpus for suggestion mining (not available at the time of this writing).

Our approach is different from the existing prior work in that we use a hybrid approach where a deep learning model is used in addition to a rule-based technique. The features extracted by rule-based approach are utilized as information sources to an LSTM network where the customer reviews are also fed into as textual input. In this way, we intend to use as many information sources as possible to improve results of a suggestion mining classifier.

3 Corpus

The corpus provided by the Semeval 2019 Suggestion Mining Challenge (Negi et al., 2019) was highly imbalanced as can be seen in Figure 1. There was a total of 8500 reviews, only 2085 of which were suggestions. The test set consisted of 824 observations.

Due to the nature of the challenge, the training set and the test set were from different domains. While the training set contained software/application reviews, test set was collected from hotel reviews. An excerpt from the training set can be seen in Table 1. The word clouds for these two datasets (Figures 2 and 3) show the difference in the most frequently used words.

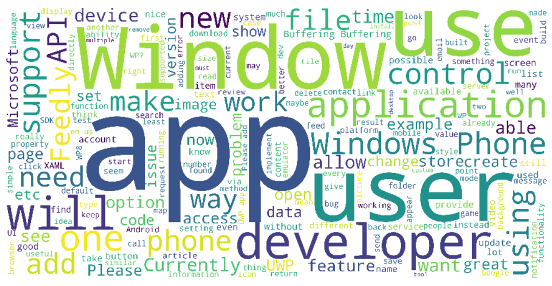


Figure 3: Word cloud of the reviews in the training set.

Review	Class
"I would like to be able to enable WP alerts be forwarded to XBox One when I am near it or manually configured for it."	Suggestion
"When you apply new policies on already existing, especially if it is related to name, all the existing credibility and market is lost."	Non-suggestion
"I find myself having to manually tab out get figures, enter them in."	Non-suggestion
"Possible solution: Route class implements IRoutePath."	Non-suggestion
"Street names color stays black and not being centered."	Non-suggestion

Table 1: Excerpt from the training set.

4 Methodology

In this section, we explain the model architecture used for the task of suggestion mining and the features utilized by the model.

4.1 Features

For suggestion mining, we used two sets of features: rule-based and model-generated from word embeddings. In this section, we describe each of these features.

4.1.1 Rule-Based Features

The rule-based features are extracted from the heuristics used in the baseline system for this challenge. Below are explanations of each of these rule-based feature.

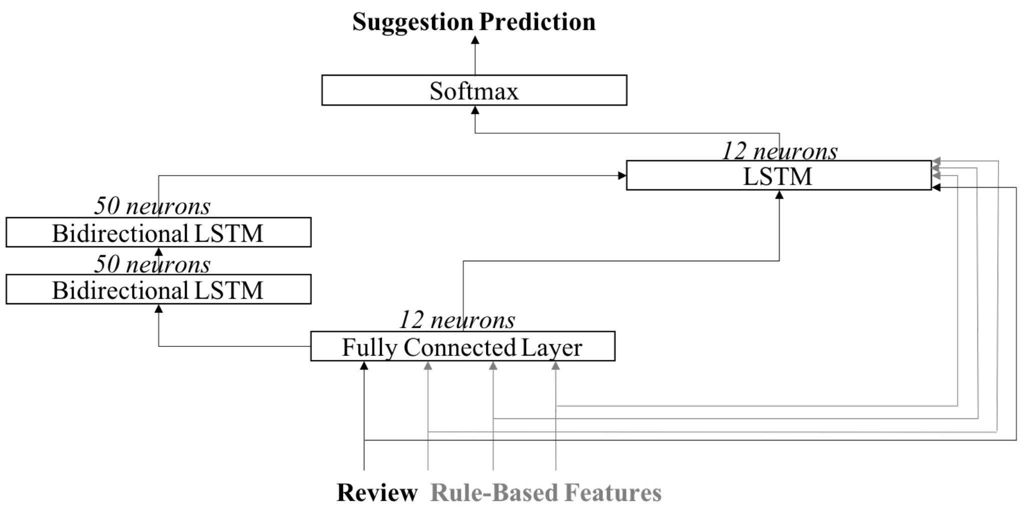


Figure 4: RNN architecture incorporating two different sources of information.

- *Rule-Based Feature 1:* the first rule-based feature is using a pattern matching algorithm based on regular expressions. This heuristic focuses on finding keywords and patterns within the input text such as '.*would \s like.*if.*' and '.*i \s wish.*'. Existence of at least one of these patterns in the review triggers a value of 1 for this feature, 0 otherwise. There are 13 patterns for this heuristic.

- *Rule-Based Feature 2:* the second rule-based feature utilizes keywords without any patterns such as 'suggest', and 'recommend'. Once one of the keywords in the list is present in the given review, the rule flags a 1 value indicating that the review contains a suggestion. There are 17 keywords in total.

- *Rule-Based Feature 3:* the third rule-based feature relies on part of speech tags. There are two part of speech tags that this heuristic is looking for (i.e., MD and VB) to be present in the tagged review to come to the conclusion that the given review is a suggestion.

4.1.2 Word Embeddings

Recurrent neural networks requires a mechanism to convert textual input to numerical vectors to be able to perform computations. To this end, we used pre-trained word embeddings where each word in the embedding table has a vector of size 100. In this study Glove embeddings is used which was trained on Wikipedia 2014 and Gigaword 5 corpora (Pennington et al., 2014).

4.2 RNN Architecture

As part of the RNN architecture, we used a fully-connected layer that takes the rule-based features and the review as the inputs. Then two bidirectional LSTM layers follow for modeling the textual input. Before the softmax layer, an LSTM layer takes the advantage of both learned representations form bidirectional layers and the rule-based features. Figure 4 depicts the architecture of the RNN model.

In the bidirectional layers, we used a dropout of 0.2 and MSRA initialization (He et al., 2015) in all layers. The training set is shuffled randomly before the first epoch. During training, ADAM optimizer (Kingma and Ba, 2014) with gradient clipping is used.

4.3 Ensemble

To fuse the different approaches utilized during the modeling phase, we used an ensemble technique. This technique take the outputs of both the rule-based features and the RNN model. If one of the rule-based features classify the review as a suggestion, the ensemble concludes that the review is a suggestion. If rule-based features classify the review as a non-suggestion and RNN classifies as a suggestion, the overall ensemble labels the observation as a suggestion. Otherwise, a non-suggestion tag is used. It is important to note that, RNN is also incorporating the rule-based features in the model. As can be seen in Figure 4, two main sources of information are fed into the model.

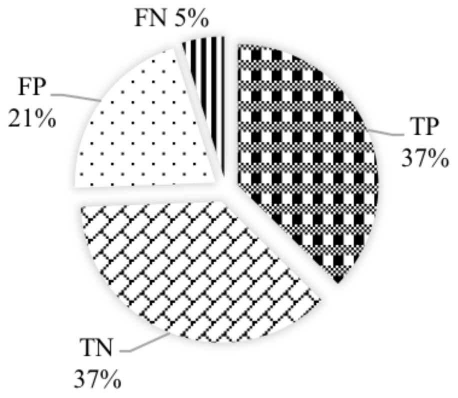


Figure 5: Pie chart showing false / true positives / negatives in the final predictions on the test set.

"and I was woken by the early morning firing up of the local bus service (a courtyard-facing room is essential unless you have industrial strength earplugs.)..."
"Don't eat breakfast in the restaurant, too costly."
"Look around the same area for another hotel."
"Avoid these rooms - it is very clear why they do not have a photograph of them on their web site."

Table 2: Samples from true positives.

5 Results

In this section, we report the results for the test set as well as discussion on the results.

5.1 Experimental Results

For the trial dataset, where the domain was hotel reviews and the majority baseline was 50%, the hybrid approach achieved F1 measure of 77.70%. It is important to note that, trial dataset has not been used to tune or validate the model. With the test dataset, the model obtained 74.49% F1 score where the majority baseline was 57.77%.

5.2 Discussion

Figure 5 shows the ratios of true/false positives and true/false negatives. From our investigation, we found out that the hybrid approach was useful in generalization of the model where the reviews did not have any keywords or patterns defined in the rules. Since RNN used generic pre-trained word embeddings (not specifically trained on either of the domains of the training set or the test set), generalization is expected for RNN. Some examples of such test observations can be seen in Table 2.

"Only one almost useless pillow per person though (think no thicker than a cracker) and no availability of additional bed linen as most other hotels would normally provide."
"Leaving your bedroom window open is not an option as my heavily bitten body will testify!"
"No shampoo provided in the room, Shower Gel dispensers don't work well."
"Put your towel on the wet floor or you will definitely slip."

Table 3: Samples from false positives.

An interesting finding in the results is about false positives. The trend observed in the false positives is that, the reviews that were helping other customers and giving hints *to the customers* rather than *to the service providers* were considered as suggestions by the model. Table 3 shows examples of those reviews where the ground truth considered these reviews as non-suggestions. However, they are suggestions to the receiving end of the service. This finding shows the difficulty of classifying this dataset because suggestions to customers and service providers can both be considered as suggestions (although not labeled as such in the ground truth).

6 Conclusion

Suggestion mining is a crucial task for mining social media data so that companies can focus on services that need improvement. Most of the times, obtaining labeled data in several different domains is not easy. Therefore, in this paper, we focused on domain-independent suggestion mining models where the training set and test set have reviews for different domains. To make our model robust, we utilized a hybrid approach that incorporates both rule-based features and relationships extracted by LSTM from raw text input. Instead of having to decide between rule-based approaches and deep learning, we fused the information sources in two ways. First by using external features in RNN and second by ensembling the result of RNN with rule-based features. By incorporating multiple information sources, we showed that the suggestion mining accuracies outperformed the baseline.

References

Carmen Banea, Rada Mihalcea, and Janyce Wiebe. 2008. A bootstrapping method for building subjectivity lexicons for languages with scarce resources. In *LREC*, volume 8, pages 2–764.

Caroline Brun and Caroline Hagege. 2013. Suggestion mining: Detecting suggestions for improvement in users' comments. *Research in Computing Science*, 70(79.7179):5379–62.

Ethem F Can, Aysu Ezen-Can, and Fazli Can. 2018. Multilingual sentiment analysis: An rnn-based framework for limited data. *arXiv preprint arXiv:1806.04511*.

Li Dong, Furu Wei, Yajuan Duan, Xiaohua Liu, Ming Zhou, and Ke Xu. 2013. The automated acquisition of suggestions from tweets. In *AAAI*.

Aysu Ezen-Can and Ethem F Can. 2018. Rnn for affects at semeval-2018 task 1: Formulating affect identification as a binary classification problem. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 162–166.

Andrew B Goldberg, Nathanael Fillmore, David Andrzejewski, Zhiting Xu, Bryan Gibson, and Xiaojin Zhu. 2009. May all your wishes come true: A study of wishes and how to recognize them. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 263–271. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Sapna Negi and P Buitelaar. 2017. Suggestion mining from opinionated text. *Sentiment Analysis in Social Networks*, pages 129–139.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*, volume 10.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Eric S Tellez, Sabino Miranda-Jiménez, Mario Graff, Daniela Moctezuma, Ranyart R Suárez, and Oscar S Siordia. 2017. A simple approach to multilingual polarity classification in twitter. *Pattern Recognition Letters*, 94:68–74.

Alfan Farizki Wicaksono and Sung-Hyon Myaeng. 2013. Automatic extraction of advice-revealing sentences foradvice mining from online forums. In *Proceedings of the seventh international conference on Knowledge capture*, pages 97–104. ACM.

INRIA at SemEval-2019 Task 9: Suggestion Mining Using SVM with Handcrafted Features

Ilia Markov
INRIA
Paris, France
ilia.markov@inria.fr

Eric Villemonte De la Clergerie
INRIA
Paris, France
eric.de_la_clergerie@inria.fr

Abstract

We present the INRIA approach to the suggestion mining task at SemEval 2019. The task consists of two subtasks: suggestion mining under single-domain (Subtask A) and cross-domain (Subtask B) settings. We used the Support Vector Machines algorithm trained on handcrafted features, function words, sentiment features, digits, and verbs for Subtask A, and handcrafted features for Subtask B. Our best run archived a F1-score of 51.18% on Subtask A, and ranked in the top ten of the submissions for Subtask B with 73.30% F1-score.

1 Introduction

Suggestion mining can be viewed as a task of extracting suggestions from unstructured text samples (Ramanand et al., 2010; Negi and Buitelaar, 2015). The task goes beyond the sentiment polarity detection and is useful for a variety of purposes, e.g., organizations can improve their products basing on the suggestions from online sources without the need of manually analyzing large amounts of unstructured data (Dong et al., 2013).

In this first edition of the suggestion mining SemEval task (Negi et al., 2019), two settings of the task are addressed: single-domain (or domain-specific) suggestion mining, where the training, development, and test sets belong to the same domain (in the context of this shared task, suggestion forum for Windows platform developers), and cross-domain setting, where training and development/test sets belong to different domains (training on developer suggestion forums and testing on hotel reviews). In the both domains, only explicit expressions of suggestions are considered: lexical cues of a suggestion are explicitly mentioned in the text (Negi et al., 2018).

We approach the task from a machine-learning perspective as a binary classification of given sentences into suggestion and non-suggestion classes. We propose a straightforward approach that can be applied when the availability of training/evaluation data and external linguistic resources is scarce, and evaluate it in the context of this shared task. We were particularly interested in evaluating our approach under cross-domain conditions (Subtask B), since this setting is more common in a real-word scenario of the task.

Further, we briefly describe the datasets used in the competition and focus on the configuration of our system.

2 Data

The training dataset provided by the organizers, as well as the development and test sets for Subtask A consist of explicit suggestion and non-suggestion sentences extracted from the feedback posts on the Universal Windows Platform[1], while the development and test sets for Subtask B are on a different domain: a subset of the sentiment analysis dataset of hotel reviews from the TripAdvisor website (Wachsmuth et al., 2014).

The training, development (dev.), and test datasets statistics in terms of the total number (no.) of sentences, the number of suggestion sentences, and the percentage (%) of suggestion sentences is provided in Table 1. A more detailed description of the datasets used in the shared task can be found in (Negi et al., 2019).

Dataset	Total no. of sentences	No. of suggestions	% of suggestions
Training	8,500	2,085	24.53%
Dev. (Subtask A)	592	296	50.00%
Dev. (Subtask B)	808	404	50.00%
Test (Subtask A)	833	87	10.44%
Test (Subtask B)	824	348	42.23%

Table 1: Suggestion mining datasets statistics.

[1]https://www.uservoice.com

As one can see from Table 1, the distribution of the suggestion and non-suggestion classes is balanced in the development sets, but imbalanced in the training and test data, which is closer to the usual distribution of the suggestion sentences in online reviews and forums (Asher et al., 2009; Negi and Buitelaar, 2015; Negi et al., 2018).

3 Methodology

In this section, we describe the features we used and the experimental setup of our best run.

3.1 Features

Handcrafted features Following previous studies on suggestion mining (Ramanand et al., 2010; Brun and Hagège, 2013; Negi and Buitelaar, 2015), we manually selected a list of representative keywords and patterns of a suggestion from the training and development data. It has been shown that suggestion expressions often contain modal verbs (Ramanand et al., 2010), e.g., *should*, *would*, which are included in our list. We also consider some verbs in their infinitive form, e.g., *suggest*, *recommend*, as well as other lexical cues such as comparative adjectives, e.g., *better*, *worse*. For Subtask A, we used a set of 57 handcrafted keywords and 77 keywords were used for Subtask B. Some of the keywords used for Subtask B did not contribute to the results obtained on the subtask A development data, and therefore were discarded. The number of such heuristic keywords in each sentence was used as a feature for the machine-learning algorithm.

Function words Function words are considered one of the most important stylometric features (Kestemont, 2014). We hypothesize that the distribution of function words is different for suggestion and non-suggestion sentences. The function word feature set consists of 318 English function words from the scikit-learn package (Pedregosa et al., 2011). Each function word was considered as a separate feature for Subtask A.

Sentiment features As mentioned in (Brun and Hagège, 2013; Negi et al., 2018), suggestions are usually expressed when a person is not entirely satisfied with the product. To capture this, we used the sentiment information from the NRC Word-Emotion Association Lexicon (Mohammad and Turney, 2013) focusing on words with negative polarity. The number of negative sentiment words in each sentence was used as a feature for Subtask A.

Digits We used the number of digits in a sentence as a feature for Subtask A. This feature is used to evaluate wether the language used in suggestion expressions is more "concrete" (as opposed to abstract) and digits usage can be one of such indicators. Other types of named and numeric entities we examined did not improve our results.

Verbs Following the work by Negi and Buitelaar (2015), we used the number of verbs in a sentence as a feature for Subtask A. The parts-of-speech (POS) tags were obtained using the TreeTagger software package (Schmid, 1995).

When used for Subtask B, function words, sentiment features, digits, and verbs did not improve the performance of our system.

3.2 Experimental setup

Classifier We used the scikit-learn (Pedregosa et al., 2011) implementation of the Support Vector Machines (SVM) algorithm, which is considered among the best-performing algorithms for text classification tasks in general, including when cross-domain conditions and binary classification are concerned (Markov et al., 2017), and for the suggestion mining task in particular (Negi and Buitelaar, 2015; Negi et al., 2016). We set the class_weight parameter to 'balanced' and the penalty parameter (C) to 0.01 for Subtask A and to 0.0001 for Subtask B, tuning the parameters according to the results on the development data.

Weighting scheme We used term frequency (tf) weighting scheme, i.e., the number of times a term occurs in a sentence.

Evaluation For the evaluation of our system, we conducted experiments on the development sets for Subtasks A and B measuring the results in terms of precision, recall, and F1-score for the positive class (the official metric). For training our system, we used only the data provided by the organizers: when evaluating on the development data, we trained our system on the training datasets, while when evaluating on the test data, we merged the training and Subtask A development sets.[2]

[2]Participants were prohibited from using additional hand-labeled training data of the same domain for Subtask B.

4 Results and discussion

First, we present the results in terms of precision (%), recall (%), and on F1-score for the positive class (%) obtained on the Subtask A development data. The contribution of each feature type incorporated in our system is shown through an ablation study in Table 2. The number of features (No.) is also provided.[3] The handcrafted features and function words are the most indicative features in our system (when used in isolation they achieve a F1-score of 72.76% and 69.77%, respectively), while other types of features slightly improve the performance of our system.

Features	Precision	Recall	F1-score	No.
All features	77.93	78.72	78.32	275
– handcrafted	75.00	66.89	70.71	274
Drop:	**2.93**	**11.83**	**7.61**	
– function words	71.72	71.96	71.84	4
Drop:	**6.21**	**6.76**	**6.48**	
– sentiment features	77.29	77.03	77.16	274
Drop:	**0.64**	**1.69**	**1.16**	
– digits	77.52	78.04	77.78	274
Drop:	**0.41**	**0.68**	**0.54**	
– verbs	77.67	78.72	78.19	274
Drop:	**0.26**	**0.00**	**0.13**	

Table 2: Ablation study of the feature types used for Subtask A.

The results in terms of precision, recall, and F1-score on the development sets for Subtasks A and B, as well as the official results obtained on the test sets are provided in Table 3. The results for the rule-based baseline approach proposed by the organizers are also presented.

Subtask A	**Precision**	**Recall**	**F1-score**
Baseline dev.	58.72	93.24	72.06
Our dev.	77.93	78.72	78.32
Baseline test	15.66	91.95	26.76
Our test	38.92	74.71	51.18
Subtask B	**Precision**	**Recall**	**F1-score**
Baseline dev.	72.85	81.68	77.01
Our dev.	85.42	82.67	84.03
Baseline test	68.86	78.16	73.22
Our test	73.62	72.99	73.30

Table 3: Results for the INRIA and the baseline approaches on the development (dev.) and test sets for Subtasks A and B.

[3]Note that we use function words as features (274 features), while the number of occurrences of the handcrafted keywords, sentiment features, digits, and verbs is considered as a feature (4 features in total).

Though the F1-score achieved by our system is higher than the one achieved by the official baselines in all cases, there is a considerable drop on the test sets: 27.14% F1-score drop for Subtask A and 10.73% for Subtask B. For Subtask A, the drop is mainly caused by the low precision achieved on the test set (precision of 77.93% on the development set and 38.92% on the test set).

After the evaluation period, in order to examine whether the drop in precision and the large number of false positives provided by our system on the Subtask A test set is partly related to the different distribution of classes in the development and test data – 50% and 10.44% of suggestions, respectively (see Table 1) –, we balanced the classes in the training and test sets to be in phase with each other and evaluated the impact of classes distribution on the results achieved by our system:

- *Test-like distribution*: we randomly removed positive examples from the training data so that the distribution of classes in the training set is the same as in the test set (10.44% of positive examples instead of 26.19% in the merged training and Subtask A development data).

- *Train-like distribution*: we removed negative examples from the test data so that the distribution of positive classes in the test set is the same as in the training data (26.19% instead of 10.44%).

The results for these two experiments are shown in Tables 4 and 5.[4]

Setting	Precision	Recall	F1-score
Original distribution	38.92	74.71	51.18
Test-like distribution	41.96	75.86	54.03
Gain:	**3.04**	**1.15**	**2.85**

Table 4: Results for the original and test-like distributions of positive classes.

Setting	Precision	Recall	F1-score
Original distribution	38.92	74.71	51.18
Train-like distribution	64.87	74.71	69.07
Gain:	**25.95**	**0.00**	**17.89**

Table 5: Results for the original and train-like distributions of positive classes.

[4]The result for the test-/train-like distributions was calculated as average over three experiments removing three different sets of positive/negative examples.

As one can see from Tables 4 and 5, balancing the distribution of positive classes, so that it is the same in the training and the evaluation data, enhances the performance of our system (by around 3% in the test-like setting and around 18% in the train-like setting) mainly due to the increase in precision, which indicates that the distribution of calsses should be taken into account when developing a robust suggestion mining system.

5 Conclusions

We presented the description of the best submission of the INRIA team to the suggestion mining shared task at SemEval 2019. Our approach is based on the Support Vector Machines algorithm trained on handcrafted features, function words, sentiment features, digits, and verbs for Subtask A (single-domain setting). For Subtask B (cross-domain setting), only handcrafted features are used. Our best run showed 51.18% F1-score for Subtask A and 73.30% for Subtask B. The results obtained on the test sets are lower than on the development data. Additional experiments revealed that the drop in F1-score is partly related to the different distribution of classes in the training data and in the development set used to evaluate and tune our system. In future work, we plan to improve our list of handcrafted features to make our system robust to variations in the distribution of classes and across different suggestion mining domains.

References

Nicholas Asher, Farah Benamara, and Yannick Mathieu. 2009. Appraisal of opinion expressions in discourse. *Lingvistic Investigationes*, 31:279–292.

Caroline Brun and Caroline Hagège. 2013. Suggestion mining: Detecting suggestions for improvement in users' comments. *Research in Computing Science*, 70:199–209.

Li Dong, Furu Wei, Yajuan Duan, Xiaohua Liu, Ming Zhou, and Ke Xu. 2013. The automated acquisition of suggestions from tweets. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, Bellevue, Washington, USA. AAAI Press.

Mike Kestemont. 2014. Function words in authorship attribution. From black magic to theory? In *Proceedings of the 3rd Workshop on Computational Linguistics for Literature*, pages 59–66, Gothenburg, Sweden. ACL.

Ilia Markov, Helena Gómez-Adorno, Grigori Sidorov, and Alexander Gelbukh. 2017. The winning approach to cross-genre gender identification in Russian at RUSProfiling 2017. In *FIRE 2017 Working Notes*, volume 2036, pages 20–24, Bangalore, India. CEUR-WS.org.

Saif Mohammad and Peter Turney. 2013. Crowdsourcing a word-emotion association lexicon. *Computational Intelligence*, 29:436–465.

Sapna Negi, Kartik Asooja, Shubham Mehrotra, and Paul Buitelaar. 2016. A study of suggestions in opinionated texts and their automatic detection. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 170–178, Berlin, Germany. ACL.

Sapna Negi and Paul Buitelaar. 2015. Towards the extraction of customer-to-customer suggestions from reviews. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics*, pages 2159–2167, Lisbon, Portugal. ACL.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Sapna Negi, Maarten de Rijke, and Paul Buitelaar. 2018. Open domain suggestion mining: Problem definition and datasets. *arXiv preprint arXiv:1806.02179*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jaiprakash Ramanand, Krishna Bhavsar, and Niranjan Pedanekar. 2010. Wishful thinking – finding suggestions and 'buy' wishes from product reviews. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 54–61, Los Angeles, California, USA. ACL.

Helmut Schmid. 1995. Improvements in part-of-speech tagging with an application to German. In *Proceedings of the ACL SIGDAT-Workshop*, pages 47–50, Dublin, Ireland. ACL.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, Gregor Engels, and Tsvetomira Palakarska. 2014. A review corpus for argumentation analysis. In *Proceedings of the 15th International Conference on Computational Linguistics and Intelligent Text Processing*, volume 8404, pages 115–127, Kathmandu, Nepal. Springer.

Lijunyi at SemEval-2019 Task 9: An attention-based LSTM model and ensemble of different models for suggestion mining from online reviews and forums

Junyi Li, Haiyan Ding*
School of Information Science and Engineering
Yunnan University, Yunnan, P.R. China
*Corresponding author: hyding@ynu.edu.cn

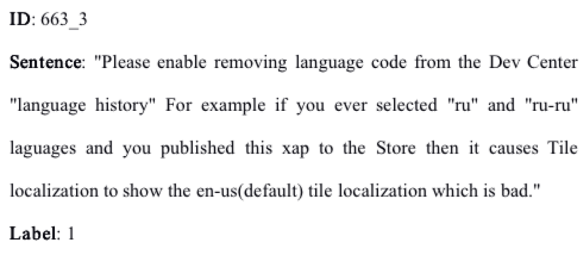
Abstract

In this paper, we describe a suggestion mining system that participated in SemEval 2019 Task 9, SubTask A - Suggestion Mining from Online Reviews and Forums. Given some suggestions from online reviews and forums that can be classified into suggestion and non-suggestion classes. In this task, we combine the attention mechanism with the LSTM model, which is the final system we submitted. The final submission achieves 14th place in Task 9, SubTask A with the accuracy of 0.6776. After the challenge, we train a series of neural network models such as convolutional neural network(CNN), TextCNN, long short-term memory(LSTM) and C-LSTM. Finally, we make an ensemble on the predictions of these models and get a better result.

1 Introduction

Suggestion mining can be defined as the extraction of suggestions from unstructured text, where the term “suggestions” refers to the expressions of tips, advice, recommendations etc(Negi et al., 2019). These suggestions largely express positive and negative sentiments towards a given entity, but also tend to contain suggestions for improving the entity. Suggestion mining remains a relatively young area compared to Sentiment Analysis, especially in the context of recent advancements in neural network based approaches for learning feature representations. In this task, suggestion mining that classified sentences into suggestion and non-suggestion classes was defined by the organizer.

In this paper, we mainly use an attention-based LSTM model(Hochreiter and Schmidhuber, 1997) for this task. The word-embedding used for all models in this task is Word2Vec. Then, the word vectors are fed into the long short-term memory (LSTM) layer. Finally, an attention mechanism(Luong et al., 2015) is added into the neural networks, and the prediction results are output via the softmax activation. What’s more, we try a number of other models (such as the TextCNN(Kim, 2014), the C-LSTM(Zhou et al., 2015) and the attention-based Bi-LSTM(Lai et al., 2015)) for comparative experiments. Furthermore we combine all of the above models to get results by soft voting.


ID: 663_3

Sentence: "Please enable removing language code from the Dev Center "language history" For example if you ever selected "ru" and "ru-ru" laguages and you published this xap to the Store then it causes Tile localization to show the en-us(default) tile localization which is bad."

Label: 1

Figure 1: An example from the SemEval 2019 Task 9 dataset

The rest of our paper is structured as follows. Section 2 introduces models. Section 3 describes data preparation. Experiments and evaluation are described in Section 4. The conclusions are drawn in Section 5.

2 Model

For this task, we use 6 models for experiments. Among these models, the attention-based LSTM models can get the best results. This model combines the attention mechanism with the LSTM. The attention mechanism is a good solution to the information vanish problem in long sequence input situations. When dealing with machine comprehension problems, the LSTM and the attention mechanism are more effective than they are used individually.

For this task, we have 4 chances to submit our result in the final submission. We use different methods that are the attention-based LSTM, C-LSTM and ensemble different models.

In this task, we not only select some single models but also use the ensemble model architecture(Sarle, 1996). The ensemble model(Kuncoro et al., 2016) architecture, shown in figure 1, is an ensemble of many single models(We call them sub models)(Dietterich, 2000). Because each sub model is independent of each other, their weights are not shared and just use the same word embedding when training each sub model. The process of the whole ensemble model is carried out model by model. First, each model is run independently, and then the result file is saved. After running all the independent models, the result files are taken out and the final result is determined by the soft vote(Rokach, 2010).

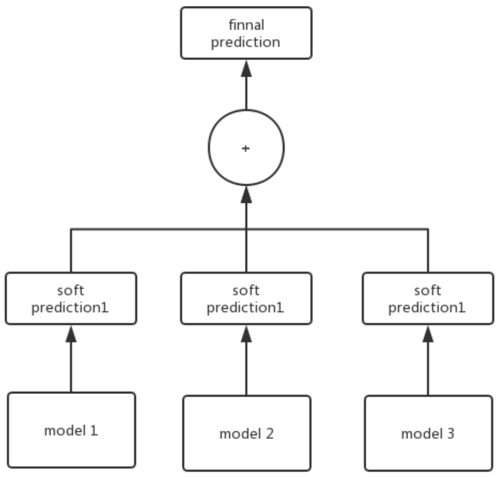


Figure 2: The architecture of the models ensemble

2.1 CNN and TextCNN

The convolutional neural network was originally used to process image data. In recent years, the application of convolutional neural networks has gradually penetrated into many different fields, such as speech recognition and natural language processing. The convolutional neural network consists of three parts. The first part is the input layer. The second part consists of *n* cyclic layers and collection layers. The third part consists of a fully connected multi-layer perceptual classifier. The difference between a cyclic neural network and a common neural network is that the convolutional neural network consists of a feature extractor with a convolutional layer and a sub-sampling layer. In the convolutional layer, one neuron is only connected to several adjacent neurons.

TextCNN is a model that uses multiple convolutional neural networks to output in tandem (Kim, 2014). In the model, the convolution window of each convolutional neural network is different in size. The convolution results obtained by convolution windows of different sizes are combined and output.

In our task, we also use the basic convolutional neural network and TextCNN to conduct experiments(Zhang and Wallace, 2015). For this task, we find that TextCNN can get a better result than a single convolutional neural network. So, we will be more inclined to choose a TextCNN model instead of a single CNN model for our task.

2.2 LSTM

Traditional recursive neural networks are ineffective when dealing with very long sentences. The LSTM (Hochreiter and Schmidhuber, 1997) model is developed to solve the gradient vanishing or exploding problems in the RNN. Currently, the LSTM is mainly used in natural language processing such as speech recognition and machine translation. Compared with the traditional RNN, an LSTM unit is added to the traditional model for judging the usefulness of information. Each unit mainly contains three gates (the forget gate, the input gate, and the output gate) and a memory cell. The system will judge the usefulness of the information after the input information is fed into an LSTM(Liu et al., 2016). Only the information that matches the rules of the algorithm will be saved, and the other information will be discarded by the forget gate.

2.3 Bi-LSTM

Single direction LSTM(Lai et al., 2015) suffers a weakness of not using the contextual information from the future tokens. Bidirectional LSTM (Bi-LSTM) exploits both the previous and future context by processing the sequence on two directions and generates two independent sequences of LSTM(Kim et al., 2016) output vectors(Liu et al., 2016). One processes the input sequence in the forward direction, while the other processes the input in the backward direction.

In this task, we also use the Bi-LSTM to get a better result(Huang et al., 2015). We select the model that can be compared with other models as comparative experiments.

2.4 C-LSTM

It has been successfully demonstrated that neural network models can achieve good results in tasks such as sentence and document classification. Convolutional neural networks (CNN) and recurrent neural networks (RNN) are two mainstream methods for this classification task (Zhou et al., 2015). At the same time, these two methods can also be used for our tasks, which use a completely different approach to understanding natural language. In this model, we combine the advantages of both CNN and RNN models and call it C-LSTM for sentence representation and text classification. C-LSTM uses CNN to extract a series of higher-level phrase representations and feeds them to the Long-Term Short-Term Memory Recurrent Neural Network (LSTM) for sentence representation(Stollenga et al., 2015). C-LSTM captures local features of phrases as well as global and temporal sentence semantics. Then, we predict the results based on the labels of the sentences (Zhou et al., 2015).

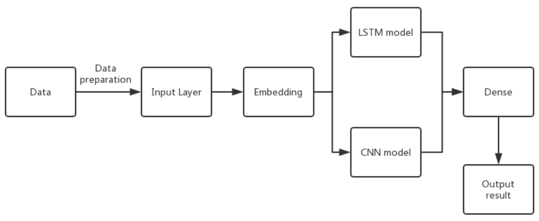


Figure 3: C-LSTM model for our task

In our experiments, the C-LSTM model is compared with a single CNN model, TextCNN, and a single LSTM, Bi-LSTM model. The results show that the C-LSTM model can achieve a better result in this task.

2.5 Attention-based LSTM model

The LSTM model can alleviate the problem of gradient vanishing, but this problem persists in long range reading comprehension contexts. The attention mechanism(Bahdanau et al., 2014) breaks the constraint on fix-length vector as the context vector, and enables the model to focus on those more helpful to outputs. After LSTM layer, we use the attention mechanism on the output vectors produced by previous layer. It is proven effective to improve the performance of our model.

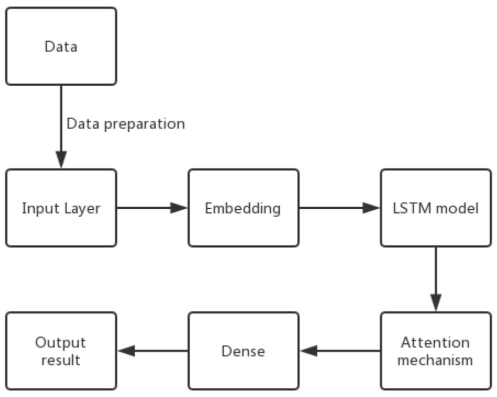


Figure 4: An attention-based LSTM model for our task

In the attention-based LSTM model, all sentences and labels are converted to word vectors by the word embedding layer. These word vectors will be fed to the LSTM layer. Subsequently, the word vector is represented as a hidden vector. Next, the attention mechanism assigns weights to each hidden vector, and the mechanism produces attention weight vectors and weighted hidden representations. Note that the weight vector is mainly obtained by calculating the similarity. An attention weight vector is generated by calculating a sentence vector matrix and a label vector matrix. The attention weight vector is then fed to the softmax layer.

The attention mechanism allows the model to retain some important hidden information when the sentence is long. In our mission, the information of sentences and tags is kept for a relatively long time. Using the standard LSTM may result in the loss of hidden information. To solve this possible problem, we have facilitated the attention-based LSTM model.

In our task, the attention mechanism(Yang et al., 2016) can get better results. We think that the attention mechanism(Vaswani et al., 2017) can improve the efficiency of task. So, we combine the attention mechanism with the LSTM model. This model can get the best results among the single models, which is the final system we submitted.

3 Data Preparation

The organizers provided training, trial, and test sets, containing 8500, 592 and 833 sentences respectively(Negi et al., 2019). Each sentence corresponds to one label, 0 or 1. Although official data is regular, we need to do a further normalization. We want to make it possible to read these sentences easily. First of all, we have completely restored the abbreviated words. For example "i'm not asking microsoft to gives permission like android so any app can take my data" will become "i am not asking microsoft to gives permission like android so any app can take my data". In this sentence "i'm" is an abbreviation. So, we found these abbreviations and restored it by creating a list.

examples	normalization
i'm	i am
doesn't	dose not
can't	can not
i'll	i will
i've	i have
...	...
i'd	i would
it's	it is

Table 1: normalization patterns

Then we noticed that it is also very important to remove some unnecessary characters, such as "!","?" etc. What's more, we find that the link to the web-page is useless for this task. So we remove all urls.

For data pre-processing, we wrote the code to realize the functions and we can improve the efficiency of our final experimental results through these data pre-processing methods.

4 Experiments and evaluation

After data pre-processing, we start the main part of the experiment. The preprocessed data is feed into our prepared model for experimentation. At the same time, we do experiments on different models to compare the test results. In the experiments, we also find that the same model will get different results under different parameter adjustments. For example, we use the C-LSTM model for experiments, and our experimental results range from 0.67 to 0.78 with different parameters in the trial data. Therefore, reasonable adjustment of parameters during the experiment is also a factor in obtaining a good experimental result.

We run each individual model 5 times and use the average as the final result of this model. In all of models, dropout parameters are changed from 0.2 to 0.6, What's more, in the LSTM model, we also select the recurrent dropout (Srivastava et al., 2014) that are set between 0.2 and 0.45. And we set epoch = 10 and batch size = 64.

In this task, we mainly select 6 models and ensemble all of these models. In the table 3 we post the F1-score and recall of the model.

Model	Recall	F1-score
CNN	0.78	0.5523
TextCNN	0.80	0.5908
LSTM	0.81	0.6104
C-LSTM	0.83	0.6222
Attention-BiLSTM	0.85	0.6610
Attention-LSTM	0.84	0.6776
ensemble models	0.82	0.6806

Table 2: Recall and F1-score for each models on task test data

5 Conclusion

In this task, we accomplish this task by integrating LSTM and attention mechanism. After competition, we try various structurally different models and an ensemble of all the models. The performance of a single model is slightly worse than the ensemble model. And there are certain differences between the different parameter results of the same model. Our results are still not as satisfying as the top teams on the leaderboard.

However, in this task, we have some problems which we can't solve. For example, we can not successfully solve the problem of data imbalance. We can not consider the problem of model optimization too much and we don't try more ways of ensemble model.

In the future, we will continue to adjust the model, improve the hardware configuration of the computer, collect more external data, and conduct more experiments to get better results. Furthermore, we will try again to solve the problem of data imbalance. We will continue to do model optimization and we will try more ways of ensemble model.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Thomas G Dietterich. 2000. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A Smith. 2016. Distilling an ensemble of greedy dependency parsers into one mst parser. *arXiv preprint arXiv:1609.07561*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Lior Rokach. 2010. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39.

Warren S Sarle. 1996. Stopped training and other remedies for overfitting. *Computing science and statistics*, pages 352–360.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Marijn F Stollenga, Wonmin Byeon, Marcus Liwicki, and Juergen Schmidhuber. 2015. Parallel multi-dimensional lstm, with application to fast biomedical volumetric image segmentation. In *Advances in neural information processing systems*, pages 2998–3006.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

MIDAS at SemEval-2019 Task 9: Suggestion Mining from Online Reviews using ULMFiT

Sarthak Anand,[3] Debanjan Mahata,[1] Kartik Aggarwal, [3] Laiba Mehnaz,[2] Simra Shahid,[2]

Haimin Zhang,[1] Yaman Kumar,[5] Rajiv Ratn Shah,[4] Karan Uppal[1]

[1]Bloomberg, USA, [2]DTU-Delhi, India, [3]NSIT-Delhi, India, [4]IIIT-Delhi, India, [5]Adobe, India,
sarthaka.ic@nsit.net.in, dmahata@bloomberg.net, kartik.mp.16@nsit.net.in,
laibamehnaz@dtu.ac.in, simrashahid_bt2k16@dtu.ac.in, hzhang449@bloomberg.net,
ykumar@adobe.com, rajivratn@iiitd.ac.in, kuppal8@bloomberg.net

Abstract

In this paper we present our approach and the system description for Sub Task A of SemEval 2019 Task 9: Suggestion Mining from Online Reviews and Forums. Given a sentence, the task asks to predict whether the sentence consists of a suggestion or not. Our model is based on Universal Language Model Fine-tuning for Text Classification. We apply various pre-processing techniques before training the language and the classification model. We further provide detailed analysis of the results obtained using the trained model. Our team ranked 10th out of 34 participants, achieving an F1 score of 0.7011. We publicly share our implementation[1].

1 Introduction and Background

Suggestion mining can be defined as the process of identifying and extracting sentences from unstructured text that contain suggestion (Negi et al., 2018). Suggestions in the form of unstructured text could be found in various social media platforms, discussion forums, review websites and blogs. They are often expressed in the form of advice, tips, recommendations, warnings, things to do, and various other forms in an explicit as well as an implicit way.

Identifying and retrieving suggestions from text can be useful in an industrial setting for enhancing a product, summarizing opinions of the consumers, giving recommendations and as an aid in decision making process (Jijkoun et al., 2010). For normal users of online platforms it could help in seeking advice related to general topics of interest like travel, health, food, shopping, education, and many more. Given the abundance of textual information in the Internet about a variety of topics, suggestion mining is certainly an useful task interesting to researchers working in academia as well as industry.

Most of the previous efforts in the direction of understanding online opinions and reactions have been limited to developing methods for areas like sentiment analysis and opinion mining (Medhat et al., 2014; Baghel et al., 2018; Kapoor et al., 2018; Mahata et al., 2018a,b; Jangid et al., 2018; Meghawat et al., 2018; Shah and Zimmermann, 2017). Mining and understanding suggestions can open new areas to study consumer behavior and tapping nuggets of information that could be directly linked with the development and enhancement of products (Brun and Hagege, 2013; Dong et al., 2013; Ramanand et al., 2010), improve customer experiences (Negi and Buitelaar, 2015), and aid in understanding the linguistic nuances of giving advice (Wicaksono and Myaeng, 2013).

Suggestion mining is a relatively new domain and is challenged by problems such as *ambiguity in task formulation and manual annotation*, *understanding sentence level semantics*, *figurative expressions*, *handling long and complex sentences*, *context dependency*, and *highly imbalanced class distribution*, as already mentioned by (Negi et al., 2018). Similar problems are also observed in the dataset shared by the organizers for the SemEval task, as it is obtained from a real-world application comprising of suggestions embedded in unstructured textual content.

Problem Definition - The problem of suggestion mining as presented in the SemEval 2019 Task 9 (Negi et al., 2019), is posed as a binary classification problem and could be formally stated as:

[1]https://github.com/isarth/SemEval9_MIDAS

Given a labeled dataset D of sentences, the objective of the task is to learn a classification/prediction function that can predict a label l for a sentence s, where $l \in \{suggestion, nonsuggestion\}$.

Our Contributions - Some of the contributions that we make by participating in this task are:

• To our knowledge we are the first one to use Universal Language Model Fine-tuning for Text Classification (ULMFiT) (Howard and Ruder, 2018), for the task of suggestion mining and show the effectiveness of transfer learning.

• We perform an error analysis of the provided dataset for Sub Task A, as well as the predictions made by our trained model.

Next, we give a detailed description of our system and the experiments performed by us along with explaining our results.

2 Experiments

2.1 Dataset

The dataset used in all our experiments was provided by the organizers of the task and consists of sentences from a suggestion forum annotated by humans to be a *suggestion* or a *non-suggestion*. Suggestion forums are dedicated forums used for providing suggestions on a specific product, service, process or an entity of interest. The provided dataset is collected from uservoice.com[2], and consists of feedback posts on Universal Windows Platform. Only those sentences are present in the dataset that explicitly expresses suggestions, for example - *Do try the cupcakes from the bakery next door*, instead of those that contain implicit suggestions such as - *I loved the cup cakes from the bakery next door* (Negi et al., 2018).

Label	**Train**	**Trial**
Suggestion	2085	296
Non Suggestion	6415	296

Table 1: Dataset Distribution for Sub Task A - Task 9: Suggestion Mining from Online Reviews.

For Sub Task A, the organizers shared a training and a validation dataset whose label distribution (*suggestion* or a *non-suggestion*) is presented in Table 1. The unlabeled test data on which the performance of our model was evaluated was also from the same domain. As evident from Table 1, there is a significant imbalance in the distribution of training instances that are *suggestions* and *non-suggestions*, which mimics the distributions of these classes in the real-world datasets. Although the dataset was collected from a suggestion forum and is expected to have a high occurrence of suggestions, yet the imbalance is more prominent due to the avoidance of implicit suggestions.

2.2 Dataset Preparation

Before using the provided dataset for training a prediction model, we take steps to prepare it as an input to our machine learning models. We primarily use Ekphrasis[3] for implementing our preprocessing steps. Some of the steps that we take are presented in this section.

2.2.1 Tokenization

Tokenization is a fundamental pre-processing step and could be one of the important factors influencing the performance of a machine learning model that deals with text. As online suggestion forums include wide variation in vocabulary and expressions, the tokenization process could become a challenging task. Ekphrasis ships with custom tokenizers that understands expressions found in colloquial languages often used in forums and has the ability to handle hashtags, dates, times, emoticons, besides standard tokenization of English language sentences. We also had to tokenize certain misspellings and slangs (eg. "I'm", "r:are") after carefully inspecting the provided dataset.

2.2.2 Normalization

After tokenization, a range of transformations such as word-normalization, spell correction and segmentation are applied to the extracted tokens. During word-normalization, URLs, usernames, phone numbers, date, time, currencies and special type of tokens such as hashtags, emoticons, censored words etc. are recognized and replaced by masks (eg. <date>, <hashtag>, <url>). These steps results in a reduction in the vocabulary size without the loss of informative excerpts that has signals for expressing suggestions. This was validated manually by analyzing the text after applying the different processing steps. Table 2 shows an example text snippet and its form after the application of the pre-processing steps.

[2]https://www.uservoice.com/

[3]https://github.com/cbaziotis/ekphrasis

Text Snippet before Pre-processing	**Text Snippet after Pre-processing**
ie9mobile does not do this :(	ie mobile does not do this <emsad>
For example if you want a feed for every Tumblr feed containing the hashtags " "#retail #design " "; " "http://www.tumblr .com/tagged/retail+ design""; would be a feedly feed."	For example if you want a feed for every tumblr feed containing the hashtags <hashtag>retail <hashtag>design <url>would be a feedly feed

Table 2: Text snippet from the dataset before and after applying pre-processing steps.

2.2.3 Class Imbalance

As already pointed in Section 2.1, *class imbalance* is a prevalent challenge in this domain and is reflected in the provided dataset. We use oversampling technique in order to tackle this challenge. We duplicate the training instances labeled as *suggestions* and boost their number of occurrences exactly to double the amount present in the original dataset.

3 Model Architecture Training and Evaluation

We show the effectiveness of transfer learning for the task of suggestion mining by training Universal Language Model Fine-tuning for Text Classification (ULMFiT) (Howard and Ruder, 2018). One of the main advantages of training ULMFiT is that it works very well for a small dataset as provided in the Sub Task A and also avoids the process of training a classification model from scratch. This avoids overfitting. We use the fast.ai[4] implementation of this model.

The ULMFiT model has mainly two parts, the *language model* and the *classification model*. The language model is trained on a Wiki Text corpus to capture general features of the language in different layers. We fine tune the language model on the training, validation and the evaluation data. Also, we additionally scrap around two thousand reviews from the Universal Windows Platform for training our language model. After analysis of the performance we find optimal parameters to be:

- BPTT: 70, bs: 48.
- Embedding size: 400, hidden size: 1150, num of layers: 3

We also experiment with MultinomialNB, Logistic Regression, Support Vector Machines, LSTM. For LSTM we use fasttext word embeddings[5] having 300 dimensions trained on Wikipedia corpus, for representing words.

Table 3, shows the performances of all the models that we trained on the provided training dataset. We also obtained the test dataset from the organizers and evaluated our trained models on the same. The ULMFiT model achieved the best results with a F1-score of 0.861 on the training dataset and a F1-score of 0.701 on the test dataset. Table 4 shows the performance of the top 5 models for Sub Task A of SemEval 2019 Task 9. Our team ranked 10th out of 34 participants.

Model	**F1 (train)**	**F1 (test)**
Multinomial Naive Bayes (using Count Vectorizer)	0.641	0.517
Logistic Regression (using Count Vectorizer)	0.679	0.572
SVM (Linear Kernel) (using TfIdf Vectorizer)	0.695	0.576
LSTM (128 LSTM Units)	0.731	0.591
Provided Baseline	0.720	0.267
ULMFit*	0.861	0.701

Table 3: Performance of different models on the provided train and test dataset for Sub Task A.

Ranking	**Team Name**	**Performance (F1)**
1	OleNet	0.7812
2	ThisIsCompetition	0.7778
3	m_y	0.7761
4	yimmon	0.7629
5	NTUA-ISLab	0.7488
10	**MIDAS (our team)**	**0.7011***

Table 4: Best performing models for SemEval Task 9: Sub Task A.

[4]https://docs.fast.ai/text.html

[5]https://fasttext.cc/docs/en/pretrained-vectors.html

4 Error Analysis

In this section, we analyse the performance of our best model (ULMFiT) on the training data as shown by the confusion matrix presented in Figure 1. We specially look at the predictions made by our model that falls into the categories of False Positive and False Negative, as that gives us insights into the instances which our model could not classify correctly. We also present some of the instances that we found to be wrongly labeled in the provided dataset.

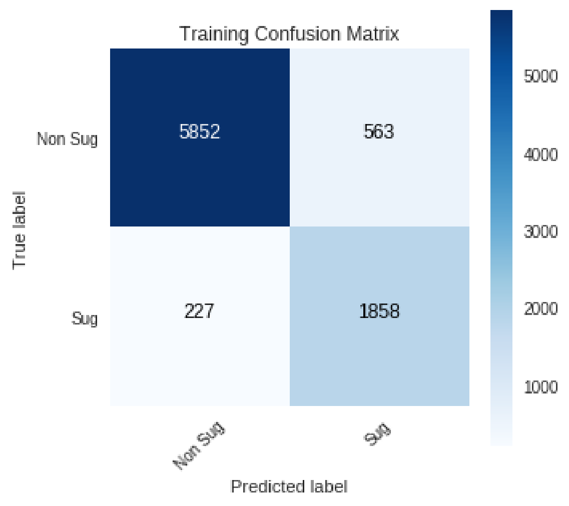


Figure 1: Confusion matrix training data

False Positives (Labeled or predicted wrongly as suggestion) Some examples that seems incorrectly labeled as suggestion in training data are given below:

- **Id 2602**: Current app extension only supports loading assets and scripts.
- **Id 3388**: One is TextCanvas for Display and Editing both Text and Inking.
- **Id 0-1747**: Unfortunately they only pull their feeds from google reader

Some examples that are incorrectly predicted by the model as suggestions are:

- **Id 1575**: That's why I'm suggesting a specialized textbox for numbers.
- **Id 1462**: If you have such limits publish them in the API docs.
- **Id 1360-2**: Adding this feature will help alot.

False Negatives (Labeled or predicted wrongly as Non Suggestion) Some examples that seems incorrectly labeled as non suggestion in the training data:

- **Id 0-1594**: Please consider adding this type of feature to feedly.
- **Id 3354**: Please support the passing of all selected files as command arguments.
- **Id 0-941**: Microsoft should provide a SDK for developers to intergate such feedback system in their Apps.

Some examples that are incorrectly predicted by the model as non-suggestions:

- **Id 0-757**: Create your own 3d library.
- **Id 834-15**: Please try again after a few minutes" in Firefox.
- **Id 4166**: I want my user to stay inside my app.

We also find that **77%** of the false positives have keywords (*want, please, add, support, would, could, should, need*), with **would** being highest i.e. around 30%.

5 Conclusion and Future Work

In this work we showed how transfer learning could be used for the task of classifying sentences extracted from unstructured text as suggestion and non-suggestions. Towards this end we train a ULMFiT model on the dataset (only Sub Task A) provided by the organizers of the SemEval 2019 Task 9 and rank 10th in the competition out of 34 participating teams.

In the future we would like to experiment and show the effectiveness of our trained model in Sub Task B where the training dataset remains the same, but the test dataset consists of suggestions from a different domain. It would be interesting to see how our model performs in predicting out-of-domain suggestions and show the ability of the ULMFiT model to fine-tune itself to a completely new domain with the already existing pre-trained model. Another interesting area would be to explore Multi Task Learning models and see how the domain of suggestion mining could get benefited by borrowing weights from models trained on other related tasks and similar tasks across different domains.

References

Nupur Baghel, Yaman Kumar, Paavini Nanda, Rajiv Ratn Shah, Debanjan Mahata, and Roger Zimmermann. 2018. Kiki kills: Identifying dangerous challenge videos from social media. *arXiv preprint arXiv:1812.00399*.

Caroline Brun and Caroline Hagege. 2013. Suggestion mining: Detecting suggestions for improvement in users' comments. *Research in Computing Science*, 70(79.7179):5379–62.

Li Dong, Furu Wei, Yajuan Duan, Xiaohua Liu, Ming Zhou, and Ke Xu. 2013. The automated acquisition of suggestions from tweets. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146.

Hitkul Jangid, Shivangi Singhal, Rajiv Ratn Shah, and Roger Zimmermann. 2018. Aspect-based financial sentiment analysis using deep learning. In *Companion of the The Web Conference 2018 on The Web Conference 2018*, pages 1961–1966. International World Wide Web Conferences Steering Committee.

Valentin Jijkoun, Wouter Weerkamp, Maarten de Rijke, Paul Ackermans, and Gijs Geleijnse. 2010. Mining user experiences from online forums: an exploration. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media*, pages 17–18.

Raghav Kapoor, Yaman Kumar, Kshitij Rajput, Rajiv Ratn Shah, Ponnurangam Kumaraguru, and Roger Zimmermann. 2018. Mind your language: Abuse and offense detection for code-switched languages. *arXiv preprint arXiv:1809.08652*.

Debanjan Mahata, Jasper Friedrichs, Rajiv Ratn Shah, and Jing Jiang. 2018a. Detecting personal intake of medicine from twitter. *IEEE Intelligent Systems*, 33(4):87–95.

Debanjan Mahata, Jasper Friedrichs, Rajiv Ratn Shah, et al. 2018b. # phramacovigilance-exploring deep learning techniques for identifying mentions of medication intake from twitter. *arXiv preprint arXiv:1805.06375*.

Walaa Medhat, Ahmed Hassan, and Hoda Korashy. 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5(4):1093–1113.

Mayank Meghawat, Satyendra Yadav, Debanjan Mahata, Yifang Yin, Rajiv Ratn Shah, and Roger Zimmermann. 2018. A multimodal approach to predict social media popularity. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 190–195. IEEE.

Sapna Negi and Paul Buitelaar. 2015. Towards the extraction of customer-to-customer suggestions from reviews. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2159–2167.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Sapna Negi, Maarten de Rijke, and Paul Buitelaar. 2018. Open domain suggestion mining: Problem definition and datasets. *arXiv preprint arXiv:1806.02179*.

Janardhanan Ramanand, Krishna Bhavsar, and Niranjan Pedanekar. 2010. Wishful thinking: finding suggestions and'buy'wishes from product reviews. In *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*, pages 54–61. Association for Computational Linguistics.

Rajiv Shah and Roger Zimmermann. 2017. *Multimodal analysis of user-generated multimedia content*. Springer.

Alfan Farizki Wicaksono and Sung-Hyon Myaeng. 2013. Automatic extraction of advice-revealing sentences foradvice mining from online forums. In *Proceedings of the seventh international conference on Knowledge capture*, pages 97–104. ACM.

NL-FIIT at SemEval-2019 Task 9: Neural Model Ensemble for Suggestion Mining

Samuel Pecar, Marian Simko, Maria Bielikova
Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovicova 2, 842 16 Bratislava, Slovakia
`{samuel.pecar, marian.simko, maria.bielikova}@stuba.sk`

Abstract

In this paper, we present neural model architecture submitted to the SemEval-2019 Task 9 competition: "Suggestion Mining from Online Reviews and Forums". We participated in both subtasks for domain specific and also cross-domain suggestion mining. We proposed a recurrent neural network architecture that employs Bi-LSTM layers and also self-attention mechanism. Our architecture tries to encode words via word representations using ELMo and ensembles multiple models to achieve better results. We performed experiments with different setups of our proposed model involving weighting of prediction classes for loss function. Our best model achieved in official test evaluation score of 0.6816 for subtask A and 0.6850 for subtask B. In official results, we achieved 12th and 10th place in subtasks A and B, respectively.

1 Introduction

Review-based portals and online forums contain plethora of user-generated text. We can consider customer reviews and inputs from online forums as an important source of novel information. These texts often contain many different opinions, which are the subject of research in area of opinion mining.

On the other hand, there can be also different types of information within these texts, such as suggestions. Unlike opinions, suggestions can appear in different parts of text and also appear more sparsely. Suggestion mining, as defined in this task, can be realized as standard text classification. We perform classification to two classes, which are suggestion and non-suggestion.

As presented by organizers, suggestion mining has different challenges (Negi et al., 2019):

- Class imbalance - suggestions appear very sparsely in reviews and forums and most of the samples are negatively sampled,
- Figurative expressions - expression can be often found in social networks but it is not always in form of suggestion,
- Context dependency - some sentences can be viewed as a suggestion, if it appears in specific domain or surrounded by specific sentences,
- Long and complex sentences - suggestions can be expressed as only small part of original sentence, which can be much longer.

Unlike opinions, suggestions can be more likely extracted also by pattern matching. We can extract suggestions by different heuristic features and keywords, such as *suggest, recommend, advise* (Negi and Buitelaar, 2015). Some works deal with domain terminology, thesaurus, linguistic parser and extraction rules (Brun and Hagege, 2013). Linguistic rules were also used for identification and extraction in sentiment expression (Viswanathan et al., 2011).

We believe that different extracted information from customer reviews and online forums can offer a valuable input for both customers and owners of products or forums and this information can be also a subject for automatic opinion summarization (Pecar, 2018).

In this paper, we present a neural network architecture consisting of different types of layers, such as embedding, recurrent, transformer or self-attention layer. We continued in our previous work on multi-level pre-processing (Pecar et al., 2018). We performed experiments with different word representations, such as ELMo, BERT or GloVe. We report results of our experiments along with error analysis of our models.

2 Model

In this SemEval task, we experimented with multiple setups based on different types of neural layers on the top of an embedding layer. In Figure 1, we show general architecture of our proposed model. We also experimented with a transformer encoder, which is described in the paragraph on encoder layer below.

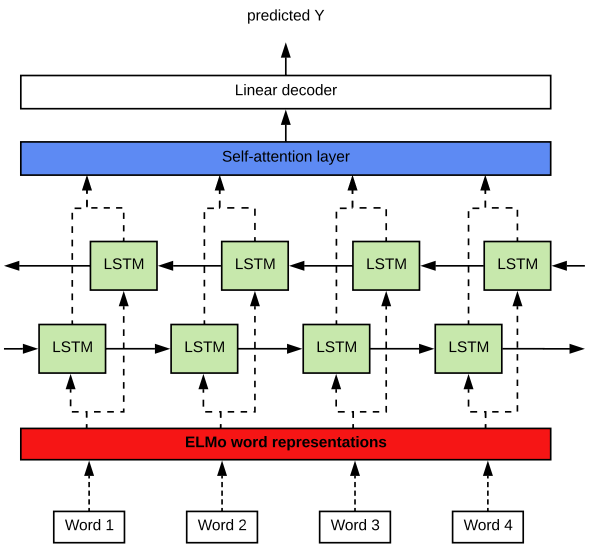


Figure 1: Proposed neural model architecture

Preprocessing We consider preprocessing of input samples as one of the most important phases in natural language processing. For user generated content, preprocessing is even more important due to noisy and ungrammatical text. We performed a study on the impact of preprocessing in our previous work (Pecar et al., 2018).

For this suggestion mining task, we used preprocessing in several stages, which were performed in order as follows:

1. text cleaning – removing all characters from not Latin alphabet, such as Cyrillic, Greek or Chinese characters,

2. character and word normalization – normalization of different use of characters and words, such as apostrophe, punctuation, date and time (e.g. ‘’ for apostrophe and ”“„ for quotation),

3. shorten phrase expanding – expanding all shorten phrases to their appropriate long form (e.g. *I'll* to *I will*),

4. expanding negations – expanding all negation forms, which appeared in short form to their appropriate long form (e.g. *aren't, won't* to *are not, will not*),

5. punctuation escaping – escaping all punctuation with spaces do separate those characters from words.

Word Representations To represent words from samples, we used deep contextualized word representations (Peters et al., 2018) also known as ELMo along with its available pre-trained model[1]. We also experimented with transformers model word representations known as BERT (Devlin et al., 2018) and its pre-trained model[2]. For language modeling in subtask B, we also experimented with GloVe embeddings (Pennington et al., 2014).

Encoder layer Word representations are fed into different encoder layers. Mostly, we used different setups of Bi-LSTM. We experimented with multiple stacked recurrent layers with different number of units within layers. In both cases (only one layer, multiple stacked layers) we used also self-attention mechanism to improve results and reduce over-fitting to the train dataset. We also tried to experiment with different attention layers and used transformer encoder (Vaswani et al., 2017) but due to very high requirements for memory, we were not able to run model with full size of this network and smaller networks produced significantly worse results.

Decoder Layer We used standard linear layer to decode output representation of recurrent layers with self-attention mechanism to class probabilities. In case of model ensemble, we needed to employ also another logarithmic softmax function for better interpretations of probabilities of samples for both classes.

Loss Function For a loss function, we experimented with the standard cross-entropy loss and the negative log likelihood loss in case a logarithmic softmax were used. We also experimented with weight setup for classes for loss contribution.

Model Ensemble Model ensemble can be considered as a useful technique to obtain better results than using only single model for predictions.

[1]https://allennlp.org/elmo
[2]https://github.com/google-research/bert

We experimented with different size of model ensemble and also two different types. In one model, we tried averaging model prediction probabilities and in second one, we used voting mechanism and predicted class with more votes.

Regularization To reduce possibilities of overfitting to train dataset, we used also dropout as a regularization technique. We used dropout on embedding layer output, on encoder output along with dropout between stacked RNN layers and also at the output of attention layer. We used different dropout probabilities in range from 0.2 to 0.6.

3 Evaluation

In this section, we briefly summarize basic information about used dataset. Later, we describe different setups of our model. Each team could submit in total 4 submission as an official results. For evaluation, binary F1 measure (F1 score over positive labels) was taken as an official results of submission.

3.1 Dataset

The dataset for suggestion mining task consists of feedback posts on Universal Windows Platform available on *uservoice.com*. The dataset contains only labels for two categories: the text is suggestion or it is not. The train dataset contains approximately 9 thousands of text samples. For validation, there were available approximately 600 samples for subtask A and 800 samples for subtask B. The size of test datasets were approximately 800 and 1000 samples for subtask A and B, respectively. More detailed information can be found in the main paper of the task (Negi et al., 2019).

3.2 Results

In Table 1, we provide basic information about setups of performed submission. For every setup, we used ELMo word representations as an embedding layer, different setups of dropout in each layer of neural network in the range from 0.3 to 0.6. Each LSTM layer has its hidden size set to 1024 units per layer. For some submissions, we also experimented with model ensemble. We took different number of models, which had the best performance on development (trial) set and used averaging predicted probabilities to get final prediction or voting mechanism and get label with more votes. In subtask A, we used 5 best trained models for voting model ensemble and 3 models for mean model ensemble. In subtask B, we used 3 best trained models for model ensemble.

In Table 1, we show also results of submitted all models in 3 measures, micro F1, macro F1 and binary F1 (F1 score over positive samples). As an official results, binary F1 measure was taken. From these results, we can observe that model ensemble can significantly help obtain better results for both subtasks.

3.3 Model ensemble results

In this section, we discuss results of each model from model ensemble in detail for both subtasks.

Table 2 shows results of each model used for model ensemble for subtask A. We can observe that the best model obtained binary F1 score 0.6609 and both types of model ensembles get better results up to 2 percents than each model separately. For mean model ensemble first 3 models were used and for voting ensemble all 5 models were used.

Table 3 shows results of each model used for model ensemble for subtask B. We can see that the best model obtained binary F1 score 0.6770 and both types of model obtained better results than each model separately. For both types of model ensemble all 3 models were used. Results for voting ensemble were not part of the official submissions.

3.4 Error analysis

We provide also error analysis of proposed model for both subtasks. We made 3 official submissions for subtask A and 2 for subtask B.

In Table 4, we show simple results from confusion matrix for subtask A and also for subtask B. For subtask A, we can observe that the main problem of our proposed models was high number of false positive labels and our models predicted presence of suggestion too often. In subtask B, there is more problematic prediction of non-suggestion labels, where number of false negative samples is much bigger. This problem can be caused also due to different distributions in training and test datasets. We also used for training dataset from a different domain, which even highlighted this problem. We used the same class weight modification for loss in subtask B as was used in subtask A.

task	submission	layers	model ensemble	micro F1	macro F1	binary F1
A	1	2 Bi-LSTM	Mean	0.9147	0.8162	0.6816
	2	2 Bi-LSTM	Voting	0.9116	0.8091	0.6696
	3	2 Bi-LSTM	None	0.9051	0.8029	0.6609
B	1	1 LSTM	Mean	0.7779	0.7567	0.6850
	2	1 LSTM	None	0.7463	0.7306	0.6656

Table 1: Official submission results in different measures

model	micro F1	macro F1	binary F1
1	0.9051	0.8029	0.6609
2	0.9050	0.8000	0.6550
3	0.9075	0.8021	0.6577
4	0.9099	0.8013	0.6543
5	0.9159	0.8061	0.6601
mean	0.9147	0.8162	0.6816
voting	0.9116	0.8091	0.6696

Table 2: Results of unsubmitted models in different measures for subtask A

model	micro F1	macro F1	binary F1
1	0.7742	0.7517	0.6770
2	0.7730	0.7446	0.6593
3	0.7463	0.7306	0.6656
mean	0.7779	0.7567	0.6850
voting	0.7574	0.7803	0.6830

Table 3: Results of unsubmitted models in different measures for subtask B

3.5 Unsubmitted models

In this section, we present results of models, which were not used to make an official submission. We experimented with these models for subtask A and also subtask B. Results can be found in Table 5 and 6. Each model in this section is used without model ensemble and we can compare results with the best models themselves. In each table, best model indicates best submitted model without any model ensemble.

The only modification used for model 1 is replacing ELMo word representation with BERT. Obtained word representations from pre-trained BERT performed much worse than ELMo representation. This fact was also observed while evaluating on development (trial) dataset.

Model 2 had a more significant modification, where LSTM encoder was replaced with transformer network (Vaswani et al., 2017). Due to high memory requirements of this model we were not able to run full encoder of the original network and used only smaller part with 6 layers and 4 head attention layers.

task	submission	TP	FP	FN	TN
A	1	76	60	11	686
	2	75	62	12	684
	3	77	69	10	677
B	1	199	34	149	442
	2	208	69	140	407

Table 4: Error analysis for submissions

As we observed in error analysis of submitted results (see Table 4), one of the significant problems was predicting too many positive labels. For all submissions, we used re-balancing class weights for loss function based on distribution in train dataset (0.6625, 2.0384). In model 3, we changed class weights to be more balanced (1.0 and 2.0). In model 4, we used completely balanced weights (1.0 and 1.0) and in model 5, we tried to change class weights to prefer negative labels (2.0 and 1.0).

model	micro F1	macro F1	binary F1
best	0.9051	0.8029	0.6609
1	0.8415	0.7214	0.5384
2	0.9123	0.7952	0.6404
3	0.9314	0.8440	0.7272
4	0.9459	0.8577	0.7457
5	0.9495	0.8479	0.7236

Table 5: Results of unsubmitted models in different measures for subtask A

For subtask B, we also experimented with use of pre-trained weights from language model. We trained language model on dataset of hotel reviews – arguana (Wachsmuth et al., 2014). Unfortunately, without fine-tuning on in-domain dataset used for classification, this model did not obtained better results. We show results of this experiment as model 1 in Table 6. We suppose that further experiment would be needed with combination of class re-balancing for loss and also fixing pre-trained weights. Since our language model was trained with GloVe embeddings, we had to use GloVe also in training for this task. Models 2 and 3 show results with change of class weight for loss function to prefer positive labels (1.0, 4.0 and 1.0, 5.0).

model	**micro F1**	**macro F1**	**binary F1**
1	0.7208	0.6698	0.5401
2	0.7961	0.7844	0.7341
3	0.8264	0.8186	0.7810

Table 6: Results of unsubmitted models in different measures for subtask B

As we showed in this section, our further experiments along with error analysis showed also significant improvement in comparison to performed official submissions. We believe further work can provide even better results, especially in combination with model ensemble.

4 Conclusions

We proposed a neural model architecture for suggestion mining. For subtask A, we employed bidirectional LSTM encoder, which consisted from 2 stacked layers followed by self attention. For subtask B, better performance proved only one layer in one direction to reduce learning process and over-fitting to train domain. Our experiments showed that pre-trained ELMo word representations performed much better than pre-trained BERT. We also performed other experiments with different setups of our architecture, which were not submitted as official results. As we showed, model ensemble can significantly improve results compared when using only single models.

To obtain better results, we would need to employ transfer learning to a much bigger extent, especially for subtask B. We could also consider further experiments with re-balancing of class weights for loss function, as we predicted too many suggestions, especially for subtask A. Another possible experiments would employ transformer network, which we were not able to fully employ due to high resource requirements. Interesting would be also employing some pattern approaches, which proved as very successful for subtask B in baseline provided by organizers. Code for our submission can be found in GitHub repository[3].

Acknowledgments

This work was partially supported by the Slovak Research and Development Agency under the contracts No. APVV-17-0267 and No. APVV SK-IL-RD-18-0004, and by the Scientific Grant Agency of the Slovak Republic grants No. VG 1/0725/19 and No. VG 1/0667/18. The authors would like to thank for financial contribution from the STU Grant scheme for Support of Young Researchers.

References

Caroline Brun and Caroline Hagege. 2013. Suggestion mining: Detecting suggestions for improvement in users' comments. *Research in Computing Science*, 70(79.7179):5379–62.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Sapna Negi and Paul Buitelaar. 2015. Towards the extraction of customer-to-customer suggestions from reviews. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2159–2167. Association for Computational Linguistics.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Samuel Pecar. 2018. Towards opinion summarization of customer reviews. In *Proceedings of ACL 2018, Student Research Workshop*, pages 1–8. Association for Computational Linguistics.

Samuel Pecar, Michal Farkaš, Marian Simko, Peter Lacko, and Maria Bielikova. 2018. NL-FIIT at IEST-2018: Emotion recognition utilizing neural networks and multi-level preprocessing. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 217–223. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language*

[3]https://github.com/SamuelPecar/NL-FIIT-SemEval19-Task9

Processing (EMNLP), pages 1532–1543. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Amar Viswanathan, Prasanna Venkatesh, Bintu G Vasudevan, Rajesh Balakrishnan, and Lokendra Shastri. 2011. Suggestion mining from customer reviews. In *AMCIS*.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, Gregor Engels, and Tsvetomira Palakarska. 2014. A review corpus for argumentation analysis. In *Proceedings of the 15th International Conference on Intelligent Text Processing and Computational Linguistics*, pages 115–127, Berlin Heidelberg New York. Springer.

NTUA-ISLab at SemEval-2019 Task 9: Mining Suggestions in the wild

Rolandos Alexandros Potamias, Alexandros Neofytou, Georgios Siolas
Intelligent Systems Laboratory,
National Technical University of Athens,
Zografou, Athens , Greece
rolpotamias@gmail.com, alex.neofytou@gmail.com
gsiolas@islab.ntua.gr

Abstract

As online customer forums and product comparison sites increase their societal influence, users are actively expressing their opinions and posting their recommendations on their fellow customers online. However, systems capable of recognizing suggestions still lack in stability. Suggestion Mining, a novel and challenging field of Natural Language Processing, is increasingly gaining attention, aiming to track user advice on online forums. In this paper, a carefully designed methodology to identify customer-to-company and customer-to-customer suggestions is presented. The methodology implements a rule-based classifier using heuristic, lexical and syntactic patterns. The approach ranked at 5^{th} and 1^{st} position, achieving an f1-score of 0.749 and 0.858 for SemEval-2019/Suggestion Mining sub-tasks A and B, respectively. In addition, we were able to improve performance results by combining the rule-based classifier with a recurrent convolutional neural network, that exhibits an f1-score of 0.79 for subtask A.

1 Introduction

Nowadays, online platforms and e-commerce sites have become increasingly popular. In such online environments people are willingly sharing views, feelings and opinions about specific products and services by conveying written recommendations on several topics. The pool of advising items that float in contemporary e-commerce environments posts a critical task: how to identify, capture and extract useful information from them. Digging into these information pools of unstructured commenting dialogues about ideas and thoughts expressed by reviewers on the product, its features, components or functions can be seen as an Opinion Mining (OM) problem (Banitaan et al., 2010), with the relevant research gaining increasingly significant attention.

However, OM strategies attempt to extract the sentiment of users written passages, failing to correctly capture putative advice, especially in the presence of mixed emotions. For example, a suggestion in an online trip advising platform could contain the sentence *“If you sleep lightly, this would be a problem because of all the trams that go by - so ask for an interior room.”*. The passage conveys a negative sentiment which is impossible to grasp correctly. Confronted with this problem, computational methodologies that automatically identify and highlight clear-cut advice in written recommendations raises as a major need. Towards this end, Suggestion Mining (SM) strives to track suggestions and tips in passages. Suggestive sentences could be positive, negative or they may not contain any sentiment at all, a fact that makes their tracking quite puzzling (Negi and Buitelaar, 2015). In general, advice may be expressed by using either *explicit suggestions*, which propose direct and conventionalized forms of recommendations (Martínez Flor, 2005), or *implicit suggestions* where the precise recommendation is in a way veiled and should be inferred (Negi and Buitelaar, 2015).

The work reported in this paper copes mainly with explicitly expressed suggestions, ignoring indirect recommendations, such as *The best thing about the Westin, as everybody has already stated, is the location.* where, via the expressed positive response, the writer implies a recommendation about *Westin* hotel. In particular, we tackle the problem of detecting suggestive comments by carefully devising a number of heuristic features to represent various lexical and imperative patterns that are indicative of explicit advice. The basic contributions of the proposed SM approach are:

- Implementation of a robust and extended identifier of imperative, prompting and solicitation clauses in sentences that are mostly related with advice and suggestions.

- Extension and elaboration on existing dictionaries and respective lexical patterns that are indicative of advice and suggestions, focusing on reviews related to electronics and hotels.
- Composition of a fully equipped and highly accurate suggestion mining framework that compares and outperforms other related approaches.

2 Related Work

Even if SM is still in its youth, few related studies aim towards the extraction of suggestions. Goldberg et al. (2009) introduced a pattern-based approach to capture the desires of customers regarding company suggestions, utilizing a Support Vector Machines (SVM) classifier. A similar study, conducted by Ramanand et al. (2010), on extracting suggestions and purchase desires for product services is founded on a pattern-based approach and the devise of respective rules. The first study that focused on the extraction of exact suggestions from product reviews and utilized feature-based classification techniques is reported in (Brun and Hagege, 2013). The same line of research is followed in Negi and Buitelaar (2015) where, an SVM approach is employed to classify heuristically devised sequential features. In a follow-up work (Negi et al., 2016) it was demonstrated that the utilization of deep learning techniques, such as Convolutional Neural Network (CNN) and Long-Short-Term-Memory (LSTM) architectures, may improve prediction performance. In a similar setting, in Golchha et al. (2018) a hybrid deep learning framework is introduced, which is composed by both recurrent and convolutional network architectures, coupled with linguistic features. A different approach is proposed in Gottipati et al. (2018) that exploits suggestions on student feedback comments utilizing a decision tree classification approach.

3 Experimental Setup

3.1 Dataset

The SemEval-2019/SM task 9 (Negi et al., 2019), targets the handling of different suggestion forms. It provides a 9K training dataset with sentences related to customer suggestions that were extracted from the "Universal Windows Platform (available from `uservoice.com`).The provided dataset is imbalanced as only 26% of cases are classified as suggestions. The provided test dataset for subtask A was also imbalanced as it contains only 87 suggestive instances from a total of 833 sentences. Data for subtask B was extracted from TripAdvisor forum[1], and carries different aspects of advice between customers. The provided test set contains 824 sample instances, 348 of which indicate explicit suggestions.

3.2 Preprocessing: *To do or not to do?*

Suggestions often comprise phrases, expressions and lexical patterns to denote their suggestion content. Such patterns may contain several *stop-words* such as *be* or *should*. We claim that retaining stop-words is needed in order to identify special lexical patterns like for example, "be sure" or "would be nice", which are indicative of the suggestion mood and content of sentences. To this end, we keep the prepossessing step as simple as possible by just lowering uppercase letters and deleting repeated punctuation. We also apply `pyspellchecker`[2] to correct any misspelled words.

4 Method

Our approach to the SM task is founded on the careful identification of lexical patterns and the assignment of contextual importance weights to them. A respective rule-based classifier is then formed that computes the degree of a sentence's suggestion content according to the importance of the included patterns. The identified patterns for both subtasks A and B are acquired by utilizing a common dictionary of suggestion patterns and related corpora, as well as imperative featured patterns described in Section 4.1. In addition, the submitted classifier for subtask A was further improved by coupling it with a convolutional recurrent neural network (Section 5).

4.1 Detection of Imperative Forms

As a linguistic property the imperative mood is found to be deeply connected to suggestions in various studies Wicaksono and Myaeng (2012). Thorough observation and analysis of the linguistic characteristics related to both subtasks, led us

[1]`https://www.tripadvisor.com.gr/`

[2]`https://pypi.org/project/pyspellchecker/`

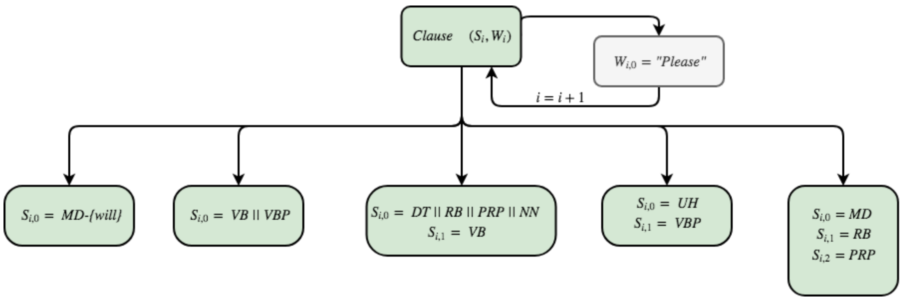


Figure 1: Imperative mood patterns based on Part of Speech Tags. Notation: $S_{i,j}$ denotes POS tag of word j in clause i and $W_{i,j}$ the mentioned word.

Pattern list $\mathbf{P_c}$ - Task A	**Pattern list $\mathbf{P_b}$** - Task B
should [not/be/take/include/start]	[do not]/[if only]
be [better]	[so/before/can/for/if] you
[that way]/[so that]/[why not]	you [will/need/can/may]
[suggestion is]/[good solution]/[the idea]	[make/be sure]/[watch out]
to allow	[go/going/asking/wishing] for
would make	would advise
[will/would] be [`Positive Sentiment`]	[will/would/could] be [`Positive Sentiment`]
[to/would/could] enable [i/would/id] [like/prefer]	be [prepared/careful/warned/forewarned] [i/would/i'd] [like/prefer]
am asking for	highly recommended
look into	[look/looking] [into/for/up/around]
make it	why not
at least	is there
we need	we need

Table 1: Sequential patterns indicating suggestive mood; left column: patterns related to subtask A; right column: patterns related to subtask B; "/" denotes logical *or*.

to extend the idea of an imperative mood detector for a task-independent and rule-based detection algorithm, able to identify the presence of prompting or solicitation clauses. The ensemble of these key features is accomplished by a mixture of rules that check both word and Part-of-Speech (POS) tag combinations, as POS tags usually avoid the sparse nature of word-based patterns. Our algorithmic approach unfolds in three steps:

- Each sentence undergoes basic preprocessing, including the replacement of each word with a tuple that contains the word and its POS tag, and enable the formation of mixed rules. In order to minimise POS errors that could lead to misclassification due to incorrect pattern recognition, a combination of the TextBlob library API[3] and the tagged Brown Corpus were utilized to assure correct word tagging.

- Each sentence is split into separate clauses that may independently indicate imperative mood, depending on their first word or verb phrase. Sentences are split at certain punctuation marks, special characters and the word 'please', as the latter provides almost absolute evidence of a following verb phrase.

- Each clause is then checked for the occurrence of certain POS tag patterns (see Figure 1), which are strong indicators of imperative mood.

4.2 Subtask A

The devised rule-based classifier assigns confidence scores to sentences on the basis of lexical-patterns organised in pre-specified categories and respective lists (detailed below). For subtask A we developed two lexical lists and one pattern-based list that vary according to their semantic content. The confidence score for each sentence is computed by a weighed additive formula that sums the hit-rates of all engaged features. A sentence is considered as suggestion if it exceeds a score of 0.15. The specific threshold is fixed after extensive experimentation on the given input dataset. Furthermore, we applied a 0.2 penalty for short phrase sentences that contain less than five words, as they

[3]https://textblob.readthedocs.io/

are considered to lack explicit suggestion content. In Table 2 we present performance results for each of these lists, as well as the prediction performance figures achieved by their combination (submitted results).

List $\mathbf{P_a^A}$. The first high rated list is composed by word features that are indicative and capture writers suggestive mood, e.g., *suggest , should , shouldn't, needs, idea, helpful, consider , allow, disallow*. Such words urge developers and supply companies to improve products and services by tracking words with directive, e.g., (*should, suggest*) or advisory content, e.g., (*idea, consider, helpful*). Sentences containing features from this list are considered as putative suggestions, assigned a 0.3 confidence score

List $\mathbf{P_b^A}$. The second list is composed by features that represent lexical patterns which, even if they do not contain individual words indicative of suggestive expressions, could be utilised when they emerge together in order to better grasp suggestive content and improve classifier's confidence. This list contains modal words, *would, could*, as well as their negations, *wouldn't, couldn't*. In addition, the list is enhanced with two developer related verbs, *create, include*. A sentence that contains both *could* and *create*, and which is rationally considered as suggestion, is: *"Read/write access to email sync settings could enable applications to create custom sync profiles (based on week-days time position etc)."*. Each hit from this list assigns a 0.1 weight to the sentence confidence score.

Patterns $\mathbf{P_c^A}$. Along with single word features we enhance our classifier with pattern-based features, elaborating on an earlier related work (Goldberg et al., 2009). To this end, we utilized several sequential lexical features as shown in Table 1. In addition, claiming that phrases like *will/would be* followed by words of positive polarity, e.g., *helpful, very very nice* are most of the times indicative of suggestions, we counted the total polarity of the three words following the phrase *will/would be*. When such patterns occur, the sentence's confidence score is increased by 0.25.

4.3 Subtask B

Following the same methodology as in subtask A we acquire a list of lexical features along with a list of weighted sequential lexical patterns. To develop lexical and n-gram patterns we extracted

Feature Set	Dev			Test		
	Prec	Rec	F1	Prec	Rec	F1
$P_a^A + P_b^A$	0.86	0.4	0.55	0.67	0.64	0.66
P_c^A	**0.87**	0.35	0.5	**0.7**	0.41	0.53
Imperative	0.83	0.23	0.36	0.67	0.21	0.32
Submitted-All	0.81	**0.71**	**0.76**	0.66	**0.86**	**0.74**

Table 2: Comparison between different lexical patterns on the development and test sets for subtask A.

Feature Set	Dev			Test		
	Prec	Rec	F1	Prec	Rec	F1
P_a^B	0.94	0.47	0.62	0.93	0.49	0.65
P_b^B	**0.98**	0.52	0.68	0.89	0.49	0.64
Imperative	0.97	0.38	0.54	**0.93**	0.36	0.52
Submitted-All	0.94	**0.78**	**0.86**	0.90	**0.82**	**0.86**

Table 3: Comparison between different lexical patterns on the development and test sets for subtask B.

the most frequent patterns from Wachsmuth et al. (2014), keeping those with the most advising content. For this subtask we consider a sentence as suggestive if it exceeds a 0.1 confidence score.

List $\mathbf{P_a^B}$. In contrast with subtask A, where suggestions appear between customers and companies, in subtask B most of the suggestions refer just to customers. Thus, we acquired a number of lexical features indicating a warning, an advice or a desire in order to capture suggestions in travel forums.

Caution words: *avoid, beware,don't, expect, remember*

Advice words: *tip, advise, advice, recommended, recommendation, suggest, suggestion, ask, bring, pick, consider, spend, expect, can*

Wish words: *please, can, hopefully, enjoying, want, wanting, prefer*

The occurrence of each of the above listed words in a sentence is assigned a weight of 0.25.

List $\mathbf{P_b^B}$. As in subtask A we used several sequential lexical patterns to identify and capture bi-grams and tri-grams with suggestive content. All lexical patterns of this list are summarized in the second column of Table 1.

5 Improving Predictions with Recurrent Convolutional Neural Networks - (R-CNN)

As simple and carefully devised lexical patterns are able to capture suggestive content in reviews and customer-to-customer conversations, we attempted to create an even more robust classifier

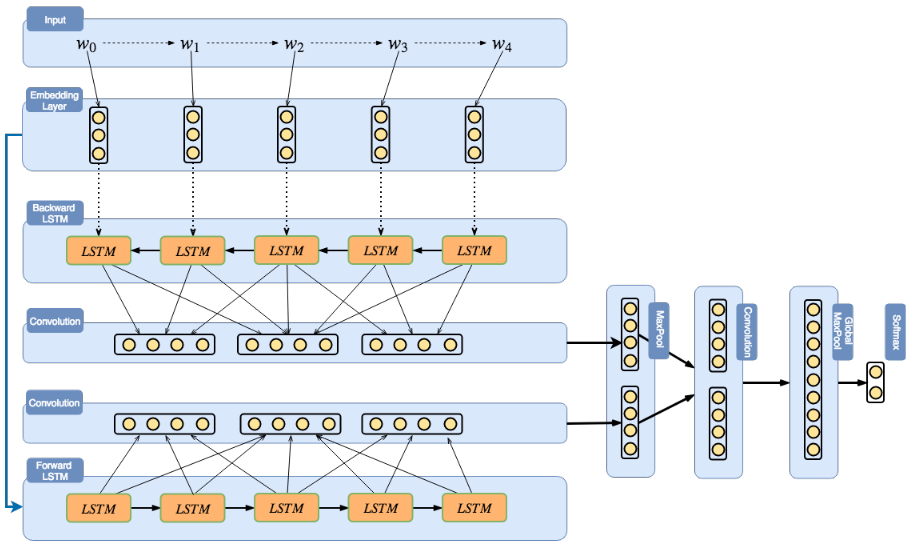


Figure 2: The recurrent convolutional neural network architecture (R-CNN).

on customer-to-companies suggestions. Thus, we coupled and enhanced our subtask A submitted classifier with a deep learning (DL) LSTM-CNN neural architecture.

5.1 Embedding Layer

DL is a very powerful classification approach, with the *Word Embeddings* machinery to be an important and necessary part for their training, especially for recurrent neural network (RNN) architectures (Mikolov et al., 2013). Word embeddings project each word to its semantic and highly dimensioned vector representation, and are induced by exhaustive training of huge corpora. In the present work we utilized 300-dimensional pre-trained Standford GloVe embeddings (Pennington et al., 2014).

5.2 Bidirectional LSTM

Due to the sequential nature of textual information neural architectures able to capture time-depended information are needed. This property is offered by RNN, and especially Long-short-term-memory (LSTM) architectures (Hochreiter and Schmidhuber, 1997). Forward LSTMs processes input information $\mathbf{x} = (x_0, x_1, ..., x_n)$ from first word x_0 to the last one x_n educing relative hidden states h_t for each time step t. However, a lot of sequential information may be hidden in long distant dependencies. Thus, to enhance forward LSTMs performance we utilized bidirectional LSTMs (Bi-LSTM) processing input from both directions, forward (from x_0 to x_n) and backward (from x_n to x_0). Thus, for a given time sample t, the hidden state h_t is defined as the concatenation of forward and backward hidden states:

$$h_t = \overrightarrow{h_t}||\overleftarrow{h_t}, \quad h_t \in R^{2*L} \tag{1}$$

where $\overrightarrow{h_t}$, $\overleftarrow{h_t}$ denotes forward and backward hidden states respectively, and $mathitL$ denotes the number of units in each LSTM cell.

5.3 Convolutional Layer

Convolutional layers are the basic component of Convolutional Neural Networks (CNN), containing m convolutional filters aiming to reduce frequency variations. The convolution layer maps input matrix $\mathbf{S} \in R^{l \times D}$ into $c \in R^{|s|+h-1}$ using convolutional filters $\mathbf{F}$ with length h, and calculating each component c_i by:

$$c_i = \sum_{k,j} (\mathbf{S}_{[i:i+h]})_{k,j} \cdot \mathbf{F_{k,j}} \tag{2}$$

Filters $\mathbf{F}$ slide over the input matrix, performing element wise product between a column slice of input s and filter matrix k, in order to produce each vector component c_i. Finally, vectors c_i aggregate over all filters, producing a feature map matrix $\mathbf{C} \in R^{m \times |s|+h-1}$, which is passed through a non-linear ReLu activation function.

5.4 Pooling Layer

To reduce the spatial size of the convolutional layer output we use a pooling layer. Pooling layers manage to tune and reduce the amount of network parameters preventing overfitting. In the current work, a MaxPooling layer is used over the convolutional layers, performing a maximum element selection of n non-overlapping intervals. Thus, MaxPooling layer results in $\mathbf{C_{pool}} \in R^{m \times \frac{|s|+h-1}{n}}$.

5.5 The Enhanced Classifier

The devised recurrent convolutional neural network (R-CNN) architecture is shown in Figure 2. Each Bi-LSTM layer, comprised by 164 units each, was fed with 300-dimensions embedding vectors. The output of Bi-LSTM is processed by two 1-d convolutional-pooling layers containing 128 ReLU activated filters of length 5. Each convolutional layer is followed by a max-pooling layer of size 5 to reduce networks parameters. For the final pooling layer we utilize a global pooling that maps all sequential patterns to a feature vector, followed by a *softmax* activated dense network. The classification result coming from the R-CNN network couples the rule-based classification (presented in Section 4.2) following a parallel fail-safe mode, that is: in the event that the softmax output exceeds a certain *confidence threshold limit* for a specific class label (as shown in Table 4), and the rule-based classification induces the opposite label, the finally assigned class label is the one induced by the R-CNN.

5.6 Training

The proposed rule-based and R-CNN coupled model was trained following a combination of train-and-trial one epoch mode. In addition, we applied several regularization techniques to prevent overfitting. In particular, we adopted Dropout (Srivastava et al., 2014; Gal and Ghahramani, 2016), empirically set to 0.3, in order to randomly deactivate 30% of recurrent neuron connections between LSTM units. Finally, to optimize network training performance we adopted Adam optimizer as proposed by Kingma and Ba (2014), using cross-entropy loss function.

6 Results

Our officially submitted results ranked in the 5^{th} and 1^{st} place for subtasks A and B, respectively (results are presented in Tables 2 and 3). Our rule-based approach that expand proposed lexical and pattern-based lexical features tend to perform significantly well for both suggestion domains, customer-to-company and customer-to-customer. As shown in Table 3, the proposed model achieves almost the same results on both development and test datasets, as well as stable precision and recall performances. The combination of lexical, sequential and imperative mood features attains 0.86 *f1-score* and robust performance over all metrics. Similar to subtask B, the submitted results for subtask A exhibit almost the same *f1-scores* on both development and test datasets, 0.76 and 0.74, respectively. However, and in contrast to subtask B, one may observe a slight unstable performance between precision and recall, as shown in Table 2. To this end, and in order to improve precision performance, we introduced the ensemble-like combination of R-CNN and rule-based classifiers (Section 5.5). As illustrated in Table 4, the combined classifier achieves increased precision figures by up to 8%, and manages to outperform related submitted systems, and attain state-of-the-art prediction scores.

Method	Test-Subtask A		
	Prec	Rec	F1
R-CNN	0.64	0.79	0.71
Submitted	0.66	**0.86**	0.74
Proposed-0.75	0.73	0.78	0.77
Proposed-0.80	0.73	0.84	0.78
Proposed-0.85	**0.74**	0.85	**0.79**

Table 4: Comparison between rule-based submitted results, R-CNN and their combination. The proposed method tested on several confidence thershold limits as defined in Section 5.5.

7 Conclusion & Future Work

The presented work, introduces and implements a pattern/rule-based classification methodology on two SM tasks that shows very good performance. In addition, we introduce and propose a combination of the pattern/rule-based classifier with a carefully devised R-CNN classifier that achieves highly accurate and state-of-the-art results. Due to the different hyperparameters and rule weight optimization needed for each dataset, our implementation is currently split into two content-specific classifiers. Therefore, a robust classifier detecting cross-domain suggestions, regardless of the input content, is an important goal to fulfill and we plan to work towards this target.

References

Shadi Banitaan, Saeed Salem, Wei Jin, and Ibrahim Aljarah. 2010. A formal study of classification techniques on entity discovery and their application to opinion mining. In *Proceedings of the 2nd international workshop on Search and mining user-generated contents*, pages 29–36. ACM.

Caroline Brun and Caroline Hagege. 2013. Suggestion mining: Detecting suggestions for improvement in users' comments. *Research in Computing Science*, 70(79.7179):5379–62.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Hitesh Golchha, Deepak Gupta, Asif Ekbal, and Pushpak Bhattacharyya. 2018. Helping each other: A framework for customer-to-customer suggestion mining using a semi-supervised deep neural network. *arXiv preprint arXiv:1811.00379*.

Andrew B Goldberg, Nathanael Fillmore, David Andrzejewski, Zhiting Xu, Bryan Gibson, and Xiaojin Zhu. 2009. May all your wishes come true: A study of wishes and how to recognize them. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 263–271. Association for Computational Linguistics.

Swapna Gottipati, Venky Shankararaman, and Jeff Rongsheng Lin. 2018. Text analytics approach to extract course improvement suggestions from students feedback. *Research and Practice in Technology Enhanced Learning*, 13(1):6.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Alicia Martínez Flor. 2005. A theoretical review of the speech act of suggesting: Towards a taxonomy for its use in flt. *Revista alicantina de estudios ingleses, No. 18 (Nov. 2005); pp. 167-187*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Sapna Negi, Kartik Asooja, Shubham Mehrotra, and Paul Buitelaar. 2016. A study of suggestions in opinionated texts and their automatic detection. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 170–178.

Sapna Negi and Paul Buitelaar. 2015. Towards the extraction of customer-to-customer suggestions from reviews. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2159–2167.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*, volume 14, pages 1532–1543.

Janardhanan Ramanand, Krishna Bhavsar, and Niranjan Pedanekar. 2010. Wishful thinking: finding suggestions and'buy'wishes from product reviews. In *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*, pages 54–61. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, Gregor Engels, and Tsvetomira Palakarska. 2014. A review corpus for argumentation analysis. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 115–127. Springer.

Alfan Farizki Wicaksono and Sung-Hyon Myaeng. 2012. Mining advices from weblogs. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2347–2350. ACM.

OleNet at SemEval-2019 Task 9: BERT based Multi-Perspective Models for Suggestion Mining

Jiaxiang Liu, Shuohuan Wang, and Yu Sun

Baidu Inc., Beijing, China,
{liujiaxiang, wangshuohuan, sunyu02}@baidu.com

Abstract

This paper describes our system participated in Task 9 of SemEval-2019: the task is focused on suggestion mining and it aims to classify given sentences into suggestion and non-suggestion classes in domain specific and cross domain training setting respectively. We propose a multi-perspective architecture for learning representations by using different classical models including Convolutional Neural Networks (CNN), Gated Recurrent Units (GRU), Feed Forward Attention (FFA), etc. To leverage the semantics distributed in large amount of unsupervised data, we also have adopted the pre-trained Bidirectional Encoder Representations from Transformers (BERT) model as an encoder to produce sentence and word representations. The proposed architecture is applied for both sub-tasks, and achieved f1-score of 0.7812 for subtask A, and 0.8579 for subtask B. We won the first and second place for the two tasks respectively in the final competition.

1 Introduction

Suggestion mining, which can be defined as the extraction of suggestions from unstructured text, where the term *suggestions* refers to the expressions of tips, advice, recommendations etc. (Negi et al., 2018). For example, *I would recommend doing the upgrade to be sure you have the best chance at trouble free operation.* and *Be sure to specify a room at the back of the hotel.* should be a suggestion for electronics and hotel separately. Collecting suggestions is an integral step of any decision making process. A suggestion mining system could extract exact suggestion sentences from a retrieved document, which would enable the user to collect suggestions from a much larger number of pages than they could manually read over a short span of time.

Suggestion mining remains a relatively young area. So far, it has usually been defined as a problem of classifying sentences of a given text into suggestion and non-suggestion classes. Mostly rule-based systems have so far been developed, and very few statistical classifiers have been proposed (Negi and Buitelaar, 2017) (Negi et al., 2016) (Negi and Buitelaar, 2015) (Brun and Hagège, 2013). A related field to suggestion mining is sentiment classification which given a sentence or a document, it should infer the sentiment polarity e.g. positive, negative, neutral. So, many classical sentiment classification systems can be used in suggestion mining like the widely used CNN-based models (Kim, 2014) or RNN-based models (Kawakami, 2008). However, there are still many challenges in this suggestion mining task. First of all, both of the subtasks suffers from severely lack of data. Second, one of the subtasks requires the model should have transferability without seeing any of the target domain data. To tackle those problems, knowledge transfer or transfer learning between domains would be desirable. In recent years, transfer learning techniques have been widely applied to solve domain adaptation problem, e.g. (Ganin et al., 2016). And in our system, considering the simplicity for training a model, we turn to taking use the power of large amount of unsupervised data for knowledge representations for both same domain and cross domain tasks.

Recently researches have shown that pre-training unsupervised language model can be very effective for learning universal language representations by leveraging large amounts of unlabeled data, e.g. the pre-trained Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018). It has shown that BERT can be fine-tuned to create state-of-the-art models for a range of NLU tasks, such as question answering

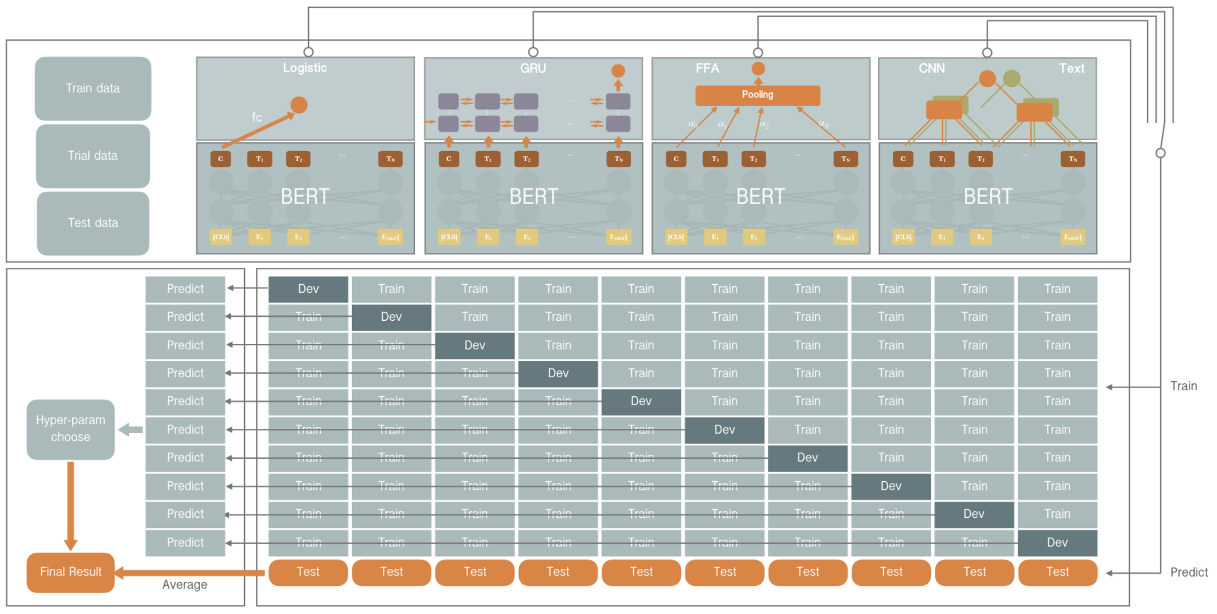


Figure 1: An overall framework and pipeline of our system for suggestion mining

and natural language inference. To further make use of the model in our task, various of different task specified layers are devised. The experiment on test datasets shows that with the devised task specified layers, a higher f1 scores can be got in both tasks, and moreover, benefiting from the large amount of unlabeled data, it is very easy to train cross domain models.

The paper is organized as follows: Section 2 describes the key models proposed for the SemEval 2019 Task 9 (Negi et al., 2019). Section 3 shows the experiment details including dataset preprocessing method, experiment configurations, threshold selection strategy and the alternatives we explored with respect to sublayers and their combination, and performances of different models. Finally, we conclude our analysis of the challenge, as well as some additional discussions of the future directions in Section 4.

2 System for Suggestion Mining

2.1 Multi-Perspective Architecture

As shown in Figure 1. our model architecture is constituted of two modules which includes a universal encoding module as either a sentence or a word encoder, and a task specified module used for suggestion classification. To fully explored the information generated by the encoder, we stack a serious of different task specified modules upon the encoder according to different perspective. Intuitively, we could use the sentence encoding directly to make a classification, to go further beyond that, as language is time-series information in essence, the time perspective based GRU cells can also be applied to model the sequence state to learn the structure for the suggestion mining task. Similarly, the spatial perspective based CNN can be used to mimic the n-gram model, as well. Moreover, we also introduce a convenient attention mechanism FFA (Raffel and Ellis, 2015) to automatically learns the combination of most important features. At last, we ensemble those models by a voting strategy as final prediction by this system. The different task specified modules will be described below.

2.2 Sentence Perspective Encoding

In the sentence encoder module, a special mark $[CLS]$ is added to the front of each sentence to help the encoder to encode all the input sentence. As a result, the output corresponds the first token is regarded as the sentence representation,

$$\begin{cases} c = E(w_t), & t = 0 \\ e_t = E(w_t), & t \in [1, T] \end{cases} \tag{1}$$

where E is the encoder module, which we user BERT in practice, c, e_t is sentence and word representation respectively, T is the total length of input sequences. We fed c into a logistic network to classify the suggestions.

2.3 Time Perspective Encoding

The Gated Recurrent Unit (GRU)(Cho et al., 2014) is famous for processing sequence data, e.g. sentences, with less parameters. In our task, we feed the e_t into GRU cells to get word representation from a time series perspective h_t,

$$\begin{aligned} z_t &= \sigma\big(e_t U^z + h_{t-1} W^z\big) \\ r_t &= \sigma\big(e_t U^r + h_{t-1} W^r\big) \\ \tilde{h}_t &= \tanh\big(e_t U^h + (r_t * h_{t-1}) W^h\big) \\ h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \\ u &= \frac{1}{T} \sum (h_t) \\ v &= \max_{1 \le t \le T} (h_t) \\ c &= [u; v] \end{aligned} \tag{2}$$

where h_t is hidden state of GRU of time step t, u is a mean pooling vector of h_t, and v is a max pooling vector of h_t. In practice, not only the h_t is used to feed into the classification layer, but the concatenated vector c is also used to train a binary classification logistic layer.

2.4 Spatial Perspective Encoding

To model the spatial connections of adjacent words, we use Convolutional Neural Network (CNN) (Kim, 2014), which is easy to implement and very fast for train. In our system, two CNN layers are stacked upon BERT model and the batch normalization (Ioffe and Szegedy, 2015) is applied in each layer. Also, the ReLu (Nair and Hinton, 2010) function is chosen as activation function. And we use max pooling to fuse the output of convolutional layers.

2.5 Attention Perspective Encoding

A recently proposed method for easier modeling of long-term dependencies is *attention* (Bahdanau et al., 2014). Attention mechanism allows for a more direct dependence between the state of the model at different points in time. Intuitively, the model with less parameters is easier to train based on small dataset, therefore, we try to use a more straight and simplified attention model, *Feed Forward Attention* (FFA) (Raffel and Ellis, 2015), which would allow it to be used to produce a single vector v from an entire sequence, the process could be formulated as follows:

$$\begin{aligned} s_t &= f(e_t) \\ \alpha_t &= \frac{exp(s_t)}{\sum_{k=1}^{T} exp(s_k)} \\ l &= \sum_{t=1}^{T} \alpha_t e_t \end{aligned} \tag{3}$$

where, f is a function mapping e_t to a unnormalized scaler s_t indicating the importance of word w_t. The l is used to make a classification to decide which input sentence is a suggestion.

But what should be noticed here is that, subtask B, whose trial data and test data are all from hotel review domain and no training data from same domain as test data is provided, is substantially a transfer learning problem. It can only learn from windows forum corpus provided in subtask A. Therefore, squeeze more cross-domain features and drop the noise is critical for subtask B. So we also introduce the hard attention mechanism (Shankar et al., 2018) :

$$l' = \sum_{\alpha \in TopK(\vec{\alpha})} \alpha h_t \tag{4}$$

we select top k important words by the attention weights α. At last the vector l and l' are used to train a binary classification logistic layer for the subtasks.

2.6 Ensemble

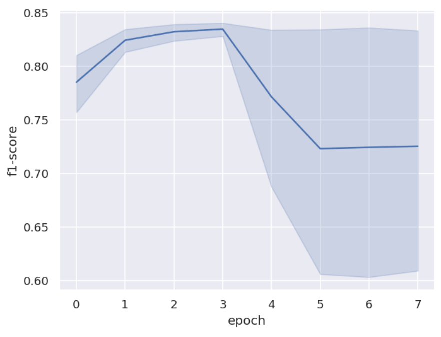
As shown in Figure 1., cross validation was adopted to ensure robustness for each model to the task 9 of SemEval-2019. In subtask A, after the 10 folds cross validation in training set has finished, the result for each fold is concatenated and used to select best classification threshold to decide a test sample from label 0 to 1. The model trained in each fold is also used to predict on test data, so the 10 test predictions is fused by mean pooling as a final prediction. Finally the simple voting method is used to fuse different model's result. In subtask B, we use the trial data as dev set to select best hyper parameters, so no cross validation is used.

3 Experiment

3.1 Dataset

The statistics of datasets provided by SemEval 2019 Task 9 are show in Table 1.

In both subtasks, no extra data are used for training models. As shown in Table 1, there are 8500


(a) Subtask A

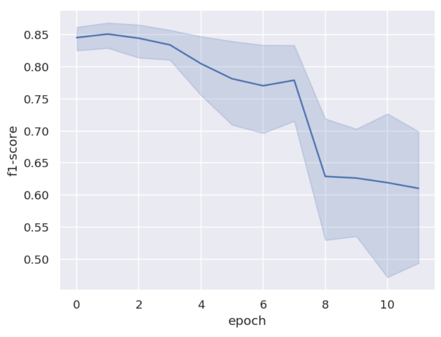

(b) Subtask B

Figure 2: Mean f1 score for trial set of different models for every epoch. In subtask A, f1 score incrementally increase and fall after the 3rd epoch, while in subtask B, f1 score of initial epoch is always surprisingly high and decreases thereafter.

Subtask A	Suggestion(%)	Non-Suggestion (%)
train	2085 (0.24)	6415 (0.75)
trial	296 (0.50)	296 (0.50)
test	87 (0.10)	746 (0.89)
Subtask B	Suggestion(%)	Non-Suggestion(%)
trial	404 (0.49)	405 (0.50)
test	348 (0.42)	476 (0.57)

Table 1: Dataset statistics for subtask A and subtask B

train examples, 592 labeled trial examples and 833 unlabeled test examples in subtask A. Different from subtask A, there are only 808 labeled trial examples, and 824 unlabeled test examples in subtask B, no training data from same domain as test data is provided. So, we use all labeled data in subtask A as the training data to do a transfer learning task to help learn subtask B. In both subtasks, the trial sets are used to help select the best model.

3.2 Details

Data Preprocessing: We use the same data cleaning method as (Cho et al., 2014), which removed the special marks. The sample is forced to *unk* if the cleaned sentence is empty.

Data augmentation method was also used in subtask A. During the error analysis procedure, we found that the model has strong tendency to learn specific terms for the task, which means the model is overfitting training data. To tackle this problem, not only dropout method is used, but also we introduce a auxiliary model to identify the importance terms according feature scores, e.g. the feature weights in a linear model. In our experiment, we use linear-kernel SVM as the auxiliary model. Specifically, we first run a linear-kernel SVM on training set. To get best performance of SVM, grid search is used to choose best hyperparameters. When finished training SVM model, the coefficients of features in the model is collected. Then, according to the value of coefficient, the most J important word are selected as key features. Finally, we replicate training samples with random dropping those important words with drop rate α to force the model to not only rely on specific terms, but also learn sentence structure of this task. In our experiment, we take J as 100, and α 0.5.

In subtask B, besides cleaning data, we combine subtask A train set and subtask A trial set to form a bigger training set. But the drop important word strategy is not applied.

Threshold choosing: As suggestion mining is introduced as a binary classification problem, choosing appropriate threshold for the logit is vital to the performance. In subtask A, a 10-fold cross validation is executed and we obtain the best threshold by calculating f1-score between the concatenated 10 validation results and training set.

In subtask B, all the data of subtask A is used as training data, and the threshold is chosen by using the subtask B trial dataset.

Empirically, the representations from BERT is universal, so after task specified fine-tuning, the performance will increases as it is show in Figure 2a. But, for the subtask B, training dataset of subtask A have a different distribution from data of subtask B. However, we assume that they still share some underlying semantics. Therefore by

Models	CV f1-score	test score
BERT-Large-Logistic	0.8522 (±0.0213)	0.7697
BERT-Large-Conv	0.8520 (±0.0231)	0.7800
BERT-Large-FFA	0.8516 (±0.0307)	0.7722
BERT-Large-GRU	0.8503 (±0.0275)	0.7725
Ensemble	–	**0.7812**

Table 2: SubtaskA models performances. CV f1-score is used to record cross validation dev set scores, and the test score is generated by trained model predicting on released labeled test data.

Models	Subtask B Trial set score	Subtask B Test set score
BERT-Large	0.8695	0.8302
BERT-Large-Conv	0.9001	0.8425
BERT-Large-FFA	0.8795	0.8409
BERT-Large-GRU	0.8796	0.8486
Ensemble	–	**0.8579**

Table 3: Subtask B models performances. We use labeled data from subtask A as training set, subtask B trial data as dev set to select best hyper-parameters, and test score is generated by trained model predicting on released labeled test data

training with subtask A data, the model should also works in the subtask B.

As shown in Figure 2a we noticed that, in subtask A, there is an obvious increasing tendency of f1 score until the 3rd epoch indicating the model have found the optimal parameters for fine-tuning. And for subtask B, which is shown in Figure 2b, best performance is always achieved in very early steps of initial epoch when fine-tuning the model and decrease all the way down, which proves that the model are learning common features cross the two different domains, but as the training process proceeds, more and more features about subtask A are learned, which cause the performance of subtask B decrease.

Learning rate tricks: Considering that the number of training dataset is too small to train a complex model, different learning rate are applied for different layers. Specifically, we apply a small learning rate for pre trained BERT layers, and a larger learning rate for new task specified layer.

3.3 Results

In the early stage of this competition, we have tried many non-BERT models, e.g. CNN (Kim, 2014), Transformer Encoder (Vaswani et al., 2017), Capsule Networks (Gong et al., 2018). However, none of the results from those models are competitive with the models based on BERT . The scores are summarized in Table 2 and 3. The result have shown that with the ensemble strategy of different models, scores of both tasks increases.

It should be noted here that, for every model of subtask A, we run 5-10 times training process repeatedly with different random seeds to ensure we can get a reliable evaluation result. For the final submission, we use the voting strategy to fuse all predictions of each model, and the ensemble results is 0.7812 and 0.8579 for subtask A and B respectively.

4 Conclusion

In this paper, we have introduced an empirical multi-perspective framework for the suggestion mining task of SemEval-2019. We propose an ensemble architecture for learning representations by using different classical models including CNN, GRU, FFA Network, etc. According to the obtained promising performances on both subtasks, we found that the pre-trained model by a large amount of unlabeled is critical for most nlp tasks, even for domain adaptation tasks without a specific neural architecture. In the future, in order to make more use of the dataset from different domains, adversarial gradient or common domain feature learning methods can be adopted along with pre-trained models to reach a better performance.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *Computer Science*.

Caroline Brun and Caroline Hagège. 2013. Suggestion mining: Detecting suggestions for improvement in users' comments. *Research in Computing Science*.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Computer Science*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Jingjing Gong, Xipeng Qiu, Shaojing Wang, and Xuanjing Huang. 2018. Information aggregation via dynamic routing for sequence encoding. *CoRR*, abs/1806.01501.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167.

Kazuya Kawakami. 2008. *Supervised sequence labelling with recurrent neural networks*. Ph.D. thesis, Ph. D. thesis, Technical University of Munich.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA. Omnipress.

Sapna Negi, Kartik Asooja, Shubham Mehrotra, and Paul Buitelaar. 2016. A study of suggestions in opinionated texts and their automatic detection. pages 170–178.

Sapna Negi and Paul Buitelaar. 2015. Towards the extraction of customer-to-customer suggestions from reviews.

Sapna Negi and Paul Buitelaar. 2017. Inducing distant supervision in suggestion mining through part-of-speech embeddings. *CoRR*, abs/1709.07403.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Sapna Negi, Maarten De Rijke, and Paul Buitelaar. 2018. Open domain suggestion mining: Problem definition and datasets.

Colin Raffel and Daniel P. W. Ellis. 2015. Feed-forward networks with attention can solve some long-term memory problems. *CoRR*, abs/1512.08756.

Shiv Shankar, Siddhant Garg, and Sunita Sarawagi. 2018. Surprisingly easy hard-attention for sequence to sequence learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 640–645.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

SSN-SPARKS at SemEval-2019 Task 9: Mining Suggestions from Online Reviews using Deep Learning Techniques on Augmented Data

Rajalakshmi S, Angel Deborah S, S Milton Rajendram, Mirnalinee T T
Department of Computer Science and Engineering
SSN College of Engineering
Chennai 603 110, Tamil Nadu, India
rajalakshmis@ssn.edu.in, angeldeborahs@ssn.edu.in
miltonrs@ssn.edu.in, mirnalineett@ssn.edu.in

Abstract

This paper describes the work on mining the suggestions from online reviews and forums. Opinion mining detects whether the comments are positive, negative or neutral, while suggestion mining explores the review content for the possible tips or advice. The system developed by SSN-SPARKS team in SemEval-2019 for task 9 (suggestion mining) uses a rule-based approach for feature selection, SMOTE technique for data augmentation and deep learning technique (Convolutional Neural Network) for classification. We have compared the results with Random Forest classifier (RF) and Multi-Layer Perceptron (MLP) model. Results show that the CNN model performs better than other models for both the subtasks.

1 Introduction

Sentiment analysis is a process of computationally identifying and categorizing the opinions from unstructured data. This can be used to identify a user's perspective of a product — positive, negative or neutral. Opinion mining is used to identify whether the product is a success in the market or not. Suggestion mining finds out ways to enhance the product to satisfy the customers.

Review texts are mainly used to identify the sentiments of the user. Besides sentiments, review texts also contain valuable information such as advice, recommendations, tips and suggestions on a variety of points of interest (Negi and Buitelaar, 2017a). These suggestions will help other customers make their choices, on the one hand, and the sellers improve their products, on the other hand.

Suggestion mining is relatively a young field of research compared to sentiment analysis. While mining for suggestions, the propositional aspects like mood, modality, sarcasm, and compound statements have to be considered. It is observed that, in some cases, grammatical properties of the sentence alone can be used to identify the label, while in other cases semantics play a significant role in label classification (Negi and Buitelaar, 2017b).

"Task 9 – Suggestion mining from online reviews and forums" has two subtasks (Negi et al., 2019). Subtask A is to classify a sentence into a suggestion or a non-suggestion. Subtask B is a cross-domain testing in which the model learned from a domain-specific dataset is used to classify dataset from a new domain. We have built classifiers using MultiLayer Perceptron (MLP), Random Forest (RF) and Convolutional Neural Network (CNN) models. However, due to the imbalance in the data, we have augmented it using Synthetic Minority Over-sampling TEchnique (SMOTE). We found that the CNN model performs better compared to RF and MLP classifiers for both the subtasks.

2 Related Work

Ramanand et al. (2010) discusses the rule-based method to find out the suggestions from the reviews. They have identified two kinds of 'wishes' viz the desire to improve the product and the desire to purchase the product. They have formulated the rules using modal verbs and certain sentence patterns. Viswanathan et al. (2011) develops an ontology-based knowledge representation for suggestion mining.

Customer-to-customer (CTC) suggestions are extracted by Negi and Buitelaar (2015) using keywords, POS tags, and imperative mood patterns. Customers feedback are analyzed using CNN and GRU network by Gupta et al. (2017). Long Short-Term Memory (LSTM) and CNN are used for sentence classification by Negi and Buitelaar (2017a).

A Linguistic-based approach is used to analyze customer experience feedback by Ordenes et al. (2014) and Brun and Hagege (2013). We have used MultiLayer Perceptron for performing task 1 and task 3 in SemEval 2018. Task 1 was to identify the affect in tweets (Angel Deborah et al., 2018) and task 3 was to identify the irony in English tweets (Rajalakshmi et al., 2018).

3 System Description

Suggestions are mined mainly for business people to improve the product or for fellow customers to detect advice (Negi et al., 2016). We have used a linguistic rule-based method for feature extraction. Data is augmented to balance the imbalance in the data. The extracted Bag of Words (BOW) features are used in MLP, RF and CNN classifier for suggestion mining.

3.1 Feature extraction

The dataset is preprocessed to remove the stop words and non-printable characters using the NLTK functions. The features from the unstructured text are extracted using parts-of-speech (POS) tag. Modal verbs (MD) and the base form of verbs (VB) are considered to be suggestion features. Since the dataset has an imbalance, we have fewer examples for the suggestion class. Hence we have added synonyms and certain keywords used in baseline to enhance the BOW feature set. Top 5 synonyms for a particular word is obtained using synsets function from wordnet and keywords such as 'suggest', 'recommend', 'add', 'extend', 'idea', 'enhance', 'helpful', and 'useful'are added to enhance the feature set.

3.2 Handling imbalanced data

Imbalance in data is a scenario where the number of observations of one class is significantly lower than those of other classes (Chawla, 2009). This problem is predominant in fraudulent transactions, rare disease identification, criminal detection and also in suggestion mining. The model developed on this dataset will be inaccurate and biased since the traditional machine learning algorithms do not consider the distribution of the classes.

Imbalance in data can be remedied using the following methods.

1. Data level resampling
   (a) Random undersampling
   (b) Random oversampling
   (c) Cluster based oversampling
   (d) Synthetic Minority Oversampling Technique (SMOTE)
   (e) Modified Synthetic Minority Oversampling technique (MSMOTE)
2. Algorithmic ensemble techniques
   (a) Bagging
   (b) Boosting: Ada boost, Gradient tree boosting, XG boost

Data augmentation can be done in data space or feature space. SMOTE algorithm is used to create augmented samples in feature space (Wong et al., 2016). A subset of minority data is used to generate similar instances, synthetically. Synthetic data are generated based on the k nearest neighbours. These synthetic data are added to the original dataset to balance it.

The procedure for balancing the minority class data (Chawla et al., 2002) is outlined in Algorithm 1. SMOTE algorithm is applied on the minority sample BOW feature vectors to generate the synthetic data.

```
Algorithm 1: SMOTE Algorithm
Input: Unbalanced dataset.
Output: Balanced dataset
begin
  1. Set the balancing ratio as auto to balance the given dataset equally.
  2. For each instance $i$ in the minority sample
     (a) Compute $k$ nearest neighbours.
     (b) For each instance $n$ in neighbour list
         i. $\mathrm{diff}_{i,n}$ = difference between $i$ and $n$.
         ii. $\mathrm{rand}$ = Generate a random number between 0 and 1.
         iii. synthetic_sample = $i + \mathrm{rand} * \mathrm{diff}_{i,n}$
         iv. Add the synthetic_sample to original dataset
end
```

3.3 Classifier algorithms

MultiLayer Perceptron, Random Forest and Convolutional Neural Network algorithms are used to build models. The augmented data is given to each of these classifiers and the models are built in Python programming environment. The results show that the CNN model performs better than MLP and RF.

3.3.1 MultiLayer Perceptron

MLP is a feedforward neural network mainly used for classification. It comprises an input layer, one or more hidden layers, and an output layer. The number of neurons in the input layer is decided by the number of features in the feature vector. The number of neurons in the output layer depends upon the number of classes. In our network, we have 5048 input neurons and 2 output neurons for suggestion/non-suggestion classification. We have used two hidden layers with 512 and 256 neurons respectively. Relu activation function is used for the input and hidden layers, while the softmax function is used for the output layer. Nadam gradient descent algorithm is used for optimization of the model.

3.3.2 Random Forest

Random forest classifier is an ensemble learning technique that uses decision tree as a basic learning algorithm. This is used to overcome the overfitting problem present in decision tree. Random forest creates a set of decision trees for randomly selected subsets of training data. It then aggregates the results of all these decision trees to make the final prediction. We have used 100 decision trees to build the random forest classifier and information gain as the measure of split criteria.

3.3.3 Convolutional Neural Network

Convolutional neural network is a deep learning technique that has already achieved remarkable results in computer vision. In text processing, deep learning techniques are used to learn the word vector representation through various neural models (Kim, 2014) and (Zhang and Wallace, 2015). We have used input embedding layer, convolutional one dimension layer with 32 filters, max-pooling layer with pool-size as 2, flatten layer, fully connected dense layer and output layer. We have added the dropout layer as 0.2 to regularize the network (Srivastava et al., 2014). The batch size of the model is set as 128 and the learning rate as 0.01. Softmax activation function is used in output layer and relu activation function is used in all other layers. Nadam algorithm is used for optimization.

3.4 Algorithm

The procedure for suggestion mining is outlined in Algorithm 2:

Algorithm 2: Suggestion Classification

```
Input: Augmented dataset.
Output: Suggestion/Non-Suggestion class labels
begin
  1. Preprocess the dataset
     (a) Separate labels and sentences.
     (b) Remove the stop words and non-ascii
         characters from the sentences using
         NLTK functions.
     (c) Perform tokenization and Parts-of-
         Speech tagging using functions of the
         NLTK toolkit.
  2. Feature selection
     (a) Identify the features using MD (Modal
         verbs) and basic form of verbs (VB).
     (b) Add the features and their synonyms
         into BoW.
     (c) Encode the features of sentences as a
         one-hot vector.
     (d) Represent the labels as a one-hot encod-
         ing of binary class in target vector.
  3. Balance the dataset using SMOTE technique.
  4. Build models (MLP, RF, CNN) with BoW
     feature vectors and target vectors.
  5. Predict the labels for the test dataset.
     (a) Preprocess the test dataset
     (b) Represent the sentences as one hot vec-
         tor with the help of BoW features of the
         training set.
     (c) Predict the labels of the test sentences by
         giving the BoW feature vector as input
         to the built model.
  6. Calculate the accuracy and F1 score.
end
```

4 Dataset

The dataset given for suggestion mining task is prepared by a study of suggestions which appeared in different domains (Negi et al., 2018). For subtask-A, suggestion forum dataset is used for training and testing. For subtask-B, suggestion forum dataset is used for training and hotel review dataset (Wachsmuth et al., 2014) is used for testing purposes. The suggestion forum training dataset has 2085 instances of suggestion class and 6415 instances of non-suggestion class. The trial test set for suggestion forum dataset has equal number of instances (296) for both classes. The trial test set for hotel review dataset has an equal number of instances (404) for both the classes.

5 Performance Evaluation

The performance of the system is measured using precision, recall and F1-score for suggestion examples alone, using formulas shown in Equations 1 to 3.

$$\text{Precision } (\text{P}_{sugg}) = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (1)$$

$$\text{Recall } (\text{R}_{sugg}) = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (2)$$

$$\text{F1 score}_{sugg} = 2 \times \frac{\text{P}_{sugg} \times \text{R}_{sugg}}{\text{P}_{sugg} + \text{R}_{sugg}} \quad (3)$$

where TP is True Positive, TN is True Negative, FP is False Positive and FN is False Negative.

The F1 score for subtask A and subtask B using various models are shown in Table 1 and 2 respectively. Results show that CNN performs slightly better than MLP and RF models. The performance of the CNN model depends on the dataset size. Hence on increasing the data, we can get better results in CNN.

F1-Score	**Value**
MLP	0.45
RF	0.4
CNN	**0.49**
Baseline	0.2676

Table 1: Performance for Subtask A

F1-Score	**Value**
MLP	0.154
CNN	**0.155**

Table 2: Performance for Subtask B

We also worked with original data as such (without balancing) and created models using MLP, RF, and CNN. The F1 score for those models are very low, as almost all the samples are classified to non-suggestion class. CNN model with hand-selected features converges in less time with the same accuracy when compared to the CNN model with pre-trained Word2Vec embeddings. We intend to further investigate the model behavior using the variations of SMOTE such as borderline SMOTE, ADASYN and MSMOTE. For subtask B, the results are very low, since we have used the model built for suggestion forum dataset to make the prediction on hotel reviews dataset. The performance can be increased by incorporating transfer learning.

6 Conclusion and Future Scope

Customers generally express their opinions about an item through online reviews, blogs, discussion forums, or social media platforms. These opinions not only contain positive or negative sentiments but also contain suggestions to improve the item or advice to other customers. We have used CNN for suggestion mining. Dataset is augmented using SMOTE technique to handle the imbalance. Rule-based approach is used for feature extraction.

The performance can be improved by extracting the features using lexicons and increasing the number of convolutional layers in CNN structure. We intend to work with variations of SMOTE algorithm for balancing data and compare the results. We would also like to investigate the performance of Recurrent Neural Network (RNN) for mining suggestions from unstructured data.

References

S Angel Deborah, S Rajalakshmi, S Milton Rajendram, and TT Mirnalinee. 2018. Ssn mlrg1 at semeval-2018 task 1: Emotion and sentiment intensity detection using rule based feature selection. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 324–328.

Caroline Brun and Caroline Hagege. 2013. Suggestion mining: Detecting suggestions for improvement in users' comments. *Research in Computing Science*, 70(79.7179):5379–62.

Nitesh V Chawla. 2009. Data mining for imbalanced datasets: An overview. In *Data mining and knowledge discovery handbook*, pages 875–886. Springer.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Deepak Gupta, Pabitra Lenka, Harsimran Bedi, Asif Ekbal, and Pushpak Bhattacharyya. 2017. Iitp at ijcnlp-2017 task 4: Auto analysis of customer feedback using cnn and gru network. *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 184–193.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Sapna Negi, Kartik Asooja, Shubham Mehrotra, and Paul Buitelaar. 2016. A study of suggestions in opinionated texts and their automatic detection. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 170–178.

Sapna Negi and P Buitelaar. 2017a. Suggestion mining from opinionated text. *Sentiment Analysis in Social Networks*, pages 129–139.

Sapna Negi and Paul Buitelaar. 2015. Towards the extraction of customer-to-customer suggestions from reviews. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2159–2167.

Sapna Negi and Paul Buitelaar. 2017b. Inducing distant supervision in suggestion mining through part-of-speech embeddings. *arXiv preprint arXiv:1709.07403*.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Sapna Negi, Maarten de Rijke, and Paul Buitelaar. 2018. Open domain suggestion mining: Problem definition and datasets. *arXiv preprint arXiv:1806.02179*.

Francisco Villarroel Ordenes, Babis Theodoulidis, Jamie Burton, Thorsten Gruber, and Mohamed Zaki. 2014. Analyzing customer experience feedback using text mining: A linguistics-based approach. *Journal of Service Research*, 17(3):278–295.

S Rajalakshmi, S Milton Rajendram, TT Mirnalinee, and S Angel Deborah. 2018. Ssn mlrg1 at semeval-2018 task 3: Irony detection in english tweets using multilayer perceptron. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 633–637.

Janardhanan Ramanand, Krishna Bhavsar, and Niranjan Pedanekar. 2010. Wishful thinking: finding suggestions and'buy'wishes from product reviews. In *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*, pages 54–61. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Amar Viswanathan, Prasanna Venkatesh, Bintu G Vasudevan, Rajesh Balakrishnan, and Lokendra Shastri. 2011. Suggestion mining from customer reviews. In *AMCIS*.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, Gregor Engels, and Tsvetomira Palakarska. 2014. A review corpus for argumentation analysis. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 115–127. Springer.

Sebastien C Wong, Adam Gatt, Victor Stamatescu, and Mark D McDonnell. 2016. Understanding data augmentation for classification: when to warp? In *2016 international conference on digital image computing: techniques and applications (DICTA)*, pages 1–6. IEEE.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

Suggestion Miner at SemEval-2019 Task 9: Suggestion Detection in Online Forum using Word Graph

Usman Ahmed[1], **Humera Liaquat**[2], **Luqman Ahmed**[3] **and Syed Jawad Hussain**[1]
[1]Department of Computer Science and IT, The University of Lahore, Islamabad, Campus
[2]Department of Computer Science, COMSAT, Islamabad, Pakistan
[3]Department of Computer Science, UET, Taxila, Pakistan
usman.ahmed@cs.uol.edu.pk
humaira.699@gmail.com
aluqman734@gmail.com
jawad.hussain@cs.uol.edu.pk

Abstract

This paper describes the suggestion miner system that participates in SemEval 2019 Task 9 - SubTask A - Suggestion Mining from Online Reviews and Forums. The system participated in the subtasks A. This paper discusses the results of our system in the development, evaluation and post evaluation. Each class in the dataset is represented as directed unweighted graphs. Then, the comparison is carried out with each class graph which results in a vector. This vector is used as features by a machine learning algorithm. The model is evaluated on, hold on strategy. The organizers randomly split (8500 instances) training set (provided to the participant in training their system) and testing set (833 instances). The test set is reserved to evaluate the performance of participants systems. During the evaluation, our system ranked 31 in the Coda Lab result of the subtask A (binary class problem). The binary class system achieves evaluation value 0.35.

1 Introduction

Suggestion mining or *identification of suggestions* within the text is a relatively new area which is gaining popularity among many private and public sector organizations, service providers and consumers/customers at large due to its number of uses. Suggestion mining, in general, refers to the extraction of tips, advise or recommendations from an unstructured text, which can lead to a number of use cases (Negi et al., 2018). Currently, suggestion mining is considered to be an intrinsic part of any decision-making process, used by different entities to get an insight into people's perspectives and improve their products or services (Negi et al., 2018), (Jijkoun et al., 2010). To get reviews and suggestions, organizations either ask them from users explicitly or extract them from online reviews, blogs, discussion forums, social media etc. This is because, these platforms are progressively gaining popularity (due to their expeditious advancements and ease of use) for obtaining public opinions towards events, brands, products, services, entities etc. Consider some examples which have been seen among many online reviews, giving useful suggestions to whom it may concern: "I would recommend doing the upgrade to be sure you have the best chance at trouble-free operation.", "Be sure to specify a room at the back of the hotel" (Negi et al., 2018), "Make sure you bring plenty of sun tan lotion-very expensive in the gift shop" (Negi and Buitelaar, 2015). We can clearly see that these reviews contain suggestions and recommendations for others to make use of a service or a product if they want to avail it at its best. Before the advent and realization of the importance and use of suggestion mining, opinion mining has been used by stakeholders to mine text with a perspective of summarizing opinions on the basis of sentiment polarities (Liu, 2012). Though identifying negative and positive sentiment distribution within a text is important from a lot of perspectives, however, identification of a suggestion oriented text would be more useful for stakeholders looking for improvements in their services or products, and also, for the services seeker and potential buyers of a product (Negi and Buitelaar, 2015). Thus, automatic identification and classification of suggestion oriented text from a large corpus of raw text is the need of the hour, as, is not feasible manually.

Some empirical analysis has been done previously for automatic suggestion mining, as, (Negi and Buitelaar, 2015) used Support Vector Machine (SVM) to identify suggestion oriented sentences within customer reviews. In another study ((Negi et al., 2016)) the authors used Neural Network architecture to classify suggestions from raw text. (Negi and Buitelaar, 2017) used Long Short Term

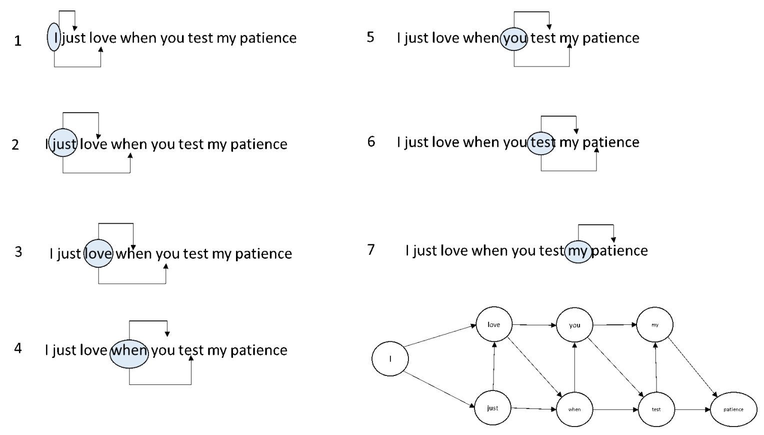

1 I just love when you test my patience
2 I just love when you test my patience
3 I just love when you test my patience
4 I just love when you test my patience
5 I just love when you test my patience
6 I just love when you test my patience
7 I just love when you test my patience
I
love
you
my
just
when
test
patience

Figure 1: Graph Construction with vicinity size 2 illustrates how the vicinity size move toward the end of the tweet; in this example, the frame is the two following words and for each word some edges and nodes are added to the graph.

Memory (LSTM) model to classify sentences as suggestions or non-suggestions. The authors, in their study trained word embeddings on suggestions (taken from WikiHow ”Tips” section) and the resulting LSTM model, showed higher performance from the previous ones. The graph-based centrality measure is also used to classify short text analysis(Ishtiaq et al., 2019). Lubna et al. proposed word sense disambigousness technique to evaluate the adverb, adjectives, and verb combination (Zafar et al., 2017).

By analysing the current suggestion mining techniques and studies ((Negi et al., 2018), (Negi and Buitelaar, 2017)), it is realized that suggestion calcification task face many overlapping challenges as with other sentence classification problems. They include: annotations of data, comprehending sentence-level semantics, making sense of figurative and sarcastic expressions, long and complex sentences (covering multiple aspects and diverse domains), catering diverse domain sentences (rather than classifying domain specific ones), imbalanced class distributions (due to the imbalanced availability of suggestion oriented text within the raw text in certain domains) etc (Negi et al., 2018).

This paper describes our proposed model for the SemEval-2019 pilot challenge for suggestion mining's Sub-Task A, i.e classify a given text as a suggestion or non-suggestion text, using same domain training and testing data. As described in the challenge highlights, the data set will belong to a particular domain i.e. suggestion forum for windows platform developers, which needs to be classified as a suggestion or non-suggestion class. For this challenge, our proposed model is a hybrid one, inspired by two previous studies ((Giannakopoulos et al., 2008) and (Maas et al., 2011)), in combination with some additional features and word graph similarity score as used by Usman et al (Ahmed et al., 2018). The word graph model used in this research is adopted from Usman et al technique of Iron detection (Ahmed et al., 2018). The detailed description of our model is given under the proposed model section. Rest of the paper includes task overview, data set description, results and evaluation, followed by discussion and conclusion.

2 Task Overview

In SemEval-2019, task 9 contains two sub-tasks A and B, for suggestion mining from a given text. The text (dataset) for this challenge which needs to be classified against each subtask is taken from two domains, i.e. suggestion forums and hotel reviews (Negi et al., 2019).

In this paper, we are focusing on sub-task A, which is, detection of a suggestion or non-suggestion text, from, the text of suggestion forums (dedicated resources used to provide suggestions for improvements in any under-discussed entity). For this task, the training and validation data sets will belong to one domain, and the details of it are covered in the dataset overview section.

3 Proposed Model

Our proposed model is a hybrid approach, in which the given text is represented as a directed unweighted word graph at first (for both the

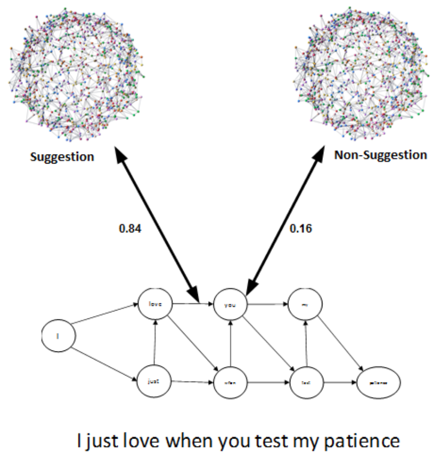


Figure 2: Graph Similarity Feature Extraction for one measure. The graph of a forum review used to compare with training data class graphs, in order to produce two numbers (depending upon the numbers of classes). These numbers will be used as a feature vector. The feature vector is provided to the trained model to predict the class of the new forum review.

classes). The edge between each word is created based on the vicinity window size, as explained in the subsection Graph construction. After graph construction, a comparison is carried out between each graph for construction of a feature vector, which is later used as an input for our machine learning algorithm. The graph is constructed based on a class assignment, which is later used to measure the similarity of a text with each class graph (suggestion or non-suggestion). For similarity measurement between the class graph and text graph, we used containment similarity, maximum common subgraph similarity and its variant compare graph in terms of similarity.

3.1 Dataset Overview

For the training, trial and evaluation dataset are provided by the organizer via Github: `https://github.com/Semeval2019Task9/Subtask-A`. The dataset for this task is annotated in two phases (Negi et al., 2019). In the first phase, crowd-sourced annotators are involved for performing the annotations, whereas in the second phase in-house expert annotators are used (Negi et al., 2019). The finalized datasets include only those sentences which explicitly express suggestions rather than those that only provide information which can be used to infer suggestions. The dataset is collected from a particular suggestion forum's reviews (uservoice.com) on universal windows platform (Negi et al., 2019). The number of sentences in the dataset is shown in Table 2.

Table 1: Feature Set

No.	Features
1	Containment Similarity UniGram
2	Containment Similarity BiGram
3	Maximum Common Subgraph Node Similarity (MCSNS)
4	Maximum Common Subgraph Edge Similarity (MCSES)
5	Maximum Common Subgraph Directed Edge Similarity (MCSDS)
6	Number of Characters
7	Number of Words

Table 2: Dataset Statistics

Class	Train set	Trial Test set	Test set
Suggestions	2085	296	87
Non-suggestions	6415	296	746

3.2 Graph Construction

For the graph construction, we consider the given text (which needs to be classified as either suggestion or non-suggestions) as a set of words, based on their vicinity. Each word is considered as a labelled node in a graph, which is joined with a directed edge, depending on the window size. For our analysis, we used a vicinity size of 2, adopted by analysing our results during the tuning phase of our model. Figure 1 illustrates the complete graph construction process of a sentence, with vicinity size 2. We can clearly see how nodes and edges are added in the graph against each word. Further, in order to check text graph similarity with the class graph, our model makes use of the containment similarity (non-normalized value), maximum common subgraph similarity and its variant compare graph.

4 Feature Engineering

4.1 Containment Similarity

Containment similarity measure is used to measure two graphs similarity by calculating the common edges between them by the number of edges of the smaller graph (Aisopos et al., 2012). Equation 1 illustrates the mathematical expression of containment similarity measure, where GT (target graph) is the text word graph, Gs (source graph) is the word graph of suggestion class. e is the edge of a word graph and the size of the graph is the

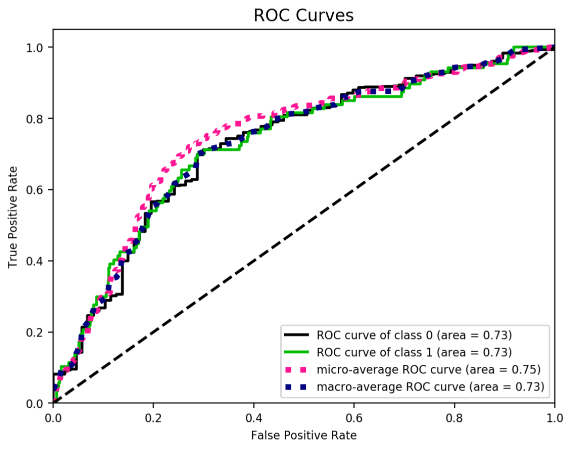


Figure 3: Precision Recall Curve of Binary Class problem

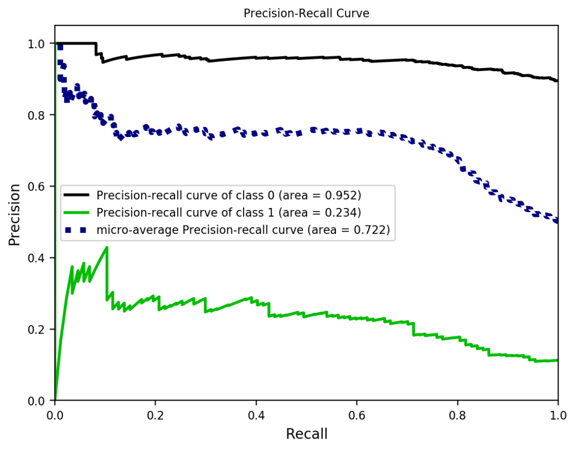


Figure 4: Precision Recall Curve of Multi Class problem

number of nodes or edges.

$$CS(G_T, G_S) = \frac{\sum_{e=G_T} \mu(e, G_S)}{min(|G_T|, |G_S|)} \quad (1)$$

4.2 Maximum Common Sub Graph

We used three variations of maximum common sub graph metric to find similarity between text graph and class graph. Equation 2 illustrates the node similarity calculation method, equation 3 illustrates the edge similarity of the two graphs where as equation 4 illustrates directed edge similarity.

$$MCSNS = \frac{MCSN(|G_T|, |G_S|)}{min(|G_T|, |G_S|)} \quad (2)$$

Maximum Common Sub graph Node Similarity (MCSNS) is the difference of target (GT) and source graph (GS).

$$MCSUES = \frac{MCSUE(|G_T|, |G_S|)}{min(|G_T|, |G_S|)} \quad (3)$$

Maximum Common Sub graph Edge Similarity (MCSUE) is the total number of edges contained in the MCS after taking the difference of target graph (GT) and source graph (GS), without considering the direction.

$$MCSDES = \frac{MCSDE(|G_T|, |G_S|)}{min(|G_T|, |G_S|)} \quad (4)$$

MAximum Common Sub graph Directed Edge Similarity (MCSDES) is the number of the edges contained in the MCS that have the same direction in the graphs. A full feature set is mentioned in Table 1.

4.3 Model Selection

To solve binary class suggestive review classification, we used Tree-based Pipeline Optimization Tool (TPOT), (Olson et al., 2016). The labelled data is given as an input to TPOT, which returns the hyper tuned model for the binary class classification problem. After close analysis it is observed that the data suffers from class imbalance problem. To handle this problem, we used SMOTE ((Cummins et al., 2017)) a Python toolbox. TPOT gives extreme gradient boosting classifier tune parameters i.e. *GradientBoostingClassifier(learning_rate=1.0, max_depth=7, max_features=0.35, min_samples_leaf=19, min_samples_split=10, n_estimators=100, subsample=1.0)* for binary classification.

5 Result Analysis and Conclusion

The model achieved rank in the Coda Lab challenge is 31, with an evaluation value of 0.34. After the release of Gold set, the model is tuned again using the same TPOT library, which is then trained and evaluated. The results are shown in the figure and .

This work describes our suggestion mining technique which is a hybrid of graph structuring and classification algorithm. Our technique uses graph similarity metrics to find similar graphs from the dataset, which later serves as an input (feature vector) to the classification algorithm. The technique generates word graphs against given reviews which are replicated throughout the dataset using graph similarity techniques. Though the results need improvement however they are convincing enough to show that use of word graphs with different vicinity window can be helpful in classifying suggestions related reviews within a domain. For further model improvements, use of different similarity metrics can be adopted as well as graph constructions using different vicinity window size can be tested.

References

Usman Ahmed, Lubna Zafar, Faiza Qayyum, and Muhammad Arshad Islam. 2018. Irony detector at semeval-2018 task 3: Irony detection in english tweets using word graph. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 581–586.

Fotis Aisopos, George Papadakis, Konstantinos Tserpes, and Theodora Varvarigou. 2012. Content vs. context for sentiment analysis: a comparative analysis over microblogs. In *Proceedings of the 23rd ACM conference on Hypertext and social media*, pages 187–196. ACM.

Chris Cummins, Pavlos Petoumenos, Zheng Wang, and Hugh Leather. 2017. Synthesizing benchmarks for predictive modeling. In *2017 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, pages 86–99. IEEE.

George Giannakopoulos, Vangelis Karkaletsis, George Vouros, and Panagiotis Stamatopoulos. 2008. Summarization system evaluation revisited: N-gram graphs. *ACM Transactions on Speech and Language Processing (TSLP)*, 5(3):5.

Asra Ishtiaq, Muhammad Arshad Islam, Muhammad Azhar Iqbal, Muhammad Aleem, and Usman Ahmed. 2019. Graph centrality based spam sms detection. In *2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, pages 629–633. IEEE.

Valentin Jijkoun, Wouter Weerkamp, Maarten de Rijke, Paul Ackermans, and Gijs Geleijnse. 2010. Mining user experiences from online forums: an exploration. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media*, pages 17–18.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics.

Sapna Negi, Kartik Asooja, Shubham Mehrotra, and Paul Buitelaar. 2016. A study of suggestions in opinionated texts and their automatic detection. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 170–178.

Sapna Negi and Paul Buitelaar. 2015. Towards the extraction of customer-to-customer suggestions from reviews. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2159–2167.

Sapna Negi and Paul Buitelaar. 2017. Inducing distant supervision in suggestion mining through part-of-speech embeddings. *arXiv preprint arXiv:1709.07403*.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Sapna Negi, Maarten de Rijke, and Paul Buitelaar. 2018. Open domain suggestion mining: Problem definition and datasets. *arXiv preprint arXiv:1806.02179*.

Randal S Olson, Nathan Bartley, Ryan J Urbanowicz, and Jason H Moore. 2016. Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 485–492. ACM.

Lubna Zafar, Muhammad Tanvir Afzal, and Usman Ahmed. 2017. Exploiting polarity features for developing sentiment analysis tool. In *EMSASW@ ESWC*.

Team Taurus at SemEval-2019 Task 9: Expert-informed pattern recognition for suggestion mining

Nelleke Oostdijk
Centre for Language Studies
Radboud University
Nijmegen, The Netherlands
N.Oostdijk@let.ru.nl

Hans van Halteren
Centre for Language Studies
Radboud University
Nijmegen, The Netherlands
hvh@let.ru.nl

Abstract

This paper presents our submissions to SemEval-2019 Task9, Suggestion Mining. Our system is one in a series of systems in which we compare an approach using expert-defined rules with a comparable one using machine learning. We target tasks with a syntactic or semantic component that might be better described by a human understanding the task than by a machine learner only able to count features. For Semeval-2019 Task 9, the expert rules clearly outperformed our machine learning model when training and testing on equally balanced testsets.

1 Introduction

In the field of natural language processing, approaches featuring machine learning (ML) nowadays predominate. These have been shown to be quite effective with a wide range of tasks, including text mining, authorship attribution, and text classification. They are particularly suited for dealing with large data volumes and are robust in the sense that they can handle quite 'noisy' data. Unlike expert-informed approaches where rules need to be hand-crafted and apply only under predefined conditions, ML approaches learn from training data and are able to extrapolate. Proponents of ML approaches tend to dismiss the enterprise of hand-crafting rules as difficult, error-prone, time-consuming, and generally ineffective as even an extensive set of complex rules is bound to be incomplete and difficult to maintain.

Over the past years we have conducted a number of studies directed at the extraction of actionable information from microblogs. These include a range of topic areas and domains, including the detection of threatening tweets (Oostdijk and van Halteren, 2013a,b), the identification of potentially contaminated food supplements in forum posts (Oostdijk et al., submitted), and topic/event detection in tweets about natural disasters (floods (Hürriyetoğlu et al., 2016), earthquakes (Hürriyetoğlu and Oostdijk, 2017; Oostdijk and Hürriyetoğlu, 2017)), about traffic flow (Oostdijk et al., 2016) and about outbreaks of the flu (Hürriyetoğlu et al., 2017). In each case we have been exploring the role of the human expert in an expert-informed pattern recognition approach and a comparable ML approach, seeking out the strengths and weaknesses of either and attempting to arrive at a superior hybrid approach.

Semeval-2019, Task 9, Suggestion Mining (Negi et al., 2019), appeared to be another task that would lend itself to human rule building. As suggestions may be phrased in many different ways, successfully recognising that an utterance contains a suggestion requires human understanding of the context. Also, the amount of available training data was quite limited so that bringing into play the human expert's knowledge of the forms that suggestions may take would be an advantage, even though the task was not too clearly defined.

As a counterpart to the human rules, we built a machine learning system. For easier comparison of the patterns suggested by the machine learner and by the human expert, the learning component was a rather simple odds-based technique which still proved competitive in VarDial2018 (van Halteren and Oostdijk, 2018). As features we used character and token n-grams as well as syntactic patterns. In addition, the machine learner was somewhat expert-informed, as it was provided with several word lists related to suggestions.

Below we first describe the rule systems built by the human expert (Section 2) and the machine learner (Section 3) in some more detail. Then we proceed to the quantitative evaluation (Section 4), followed by a qualitative analysis of the evaluation phase (Section 5). We conclude with a discussion of what we learned in this shared task (Section 6).

2 Recognition with expert rules

The expert-rule-based system we applied uses a dedicated (task-specific) lexicon and set of hand-crafted rules in order to generate search queries. The system only targets suggestions, i.e. we only specify rules for patterns that should identify suggestions. Furthermore, we chose to design and apply one and the same set of rules and lexicon for both evaluation sets.

The lexicon comprises 1256 entries. Most of the entries are single words. However, we also included some multi-word items such *in order to*, *for example*, *at least* and *be able to* as well as phrase-like items such *why not* and *how about*. The lexicon includes a few frequently observed spelling variants, for example *pls*, *plz*, *dont* and *kinda*. With each entry, part-of-speech-like information is provided (e.g. verb, adjective, adverb but also *please*, which is given its own class). Typical examples of some of the lexicon entries are:

awesome	ADJ	would	AUXwould
just	ADV	very	ADVintens
provide	Vimp	provide	Vinfin

The rule set consists of 138 finite state rules. They describe how lexical items can combine to form search queries made up of multi-word n-grams. To give some examples, `AUXwould - BE - ADVintens - ADJ` matches e.g. *would be very helpful* and *would be really awesome*; `AUXshould - ADV - Vinfin` matches e.g. *should directly see* and *should properly support*; and `PLEASE - Vimp` e.g. *please provide* and *pls fix*. Rules typically combine two to five elements (parts of speech; POS). Given that some lexical entries are multi-word items, the patterns actually describe strings up to a length of nine words (one of the strings matching the rule `AUXcould - ADV - BE - ADVintens - ADJ` is, for example, *could at the very least be even more robust*).

In previous tasks, pattern recognition using word n-grams has been proven to be very effective. Moreover, the specification of the rules and the lexicon is much easier and far less time-consuming than would be the case for an all-encompassing description, as human experts need not concern themselves with describing elements and details irrelevant to the task at hand. In this particular case, trying to recognize suggestions, we experienced the drawback of only having access to the raw text input, something that did not bother us as much with other tasks. What we found was that in order to be able to recognize the many suggestions that take the form of an imperative sentence, we somehow needed to be able to distinguish between an imperative verb form and an infinitive or present tense verb. Noting that imperative verbs tend to occur (near) sentence- or clause-initially, we tried to account for this in our rules. In some cases we could make use of the presence of a comma, semi-colon or colon, that would identify the (potential) beginning of a clause. In such cases the punctuation mark was included as an element in the rules describing imperative structures. In order to identify the beginning of a sentence, we decided to automatically insert markers in the input during the preprocessing phase. These markers would also indicate the type of sentence (interrogative `MRKques` vs declarative or imperative `MRKsint`). Here the distinction between interrogatives and other types of sentence helped in distinguishing between *do/don't/dont* as an operator (auxiliary verb) in a question and as an operator in an imperative sentence. `MRKsint - Vimp` matches e.g *Allow applications to define ...* and *Ask for a room in ...*; and `MRKsint - ADV - Vimp` matches e.g. *At least support ...* and *Just don't forget ...*

3 Recognition by machine learning

For the ML approach we tried to use quite a wide spread of feature types, namely (a) character n-grams, with n ranging from 3 to 9; (b) token unigrams; (c) token n-grams, with n ranging from 2 to 4 (in which the individual positions can be filled with the actual token, the POS tag, or the word group (see below)); (d) syntactic structure of the main clause; (e) syntactic rewrites of all constituents; (f) syntactic n-grams (i.e. selected subtrees from the complete parse tree, e.g. function and category of a mother and two daughters).

For the character n-grams and the tokens, we applied a minimal level of pre-processing. In the training data, we cleaned up misplaced quotes and commas in the provided .csv-files. In the (later) trial data and in the evaluation data, which came in an HTML-format, we also replaced SGML-entities by their character counterpart, e.g. `&quot;` was replaced by `"`.

The POS tags and syntactic structure were produced by the Stanford NLP system (Manning et al., 2014). The dependency parse labeling was then transformed to a constituency tree that conformed (as much as feasible) to our own view of English syntactic structure, being that developed in the TOSCA project (Aarts et al., 1998). All syntactic features were derived in either a fully lexicalized or fully unlexicalized version. Although we feel the syntactic features can be of considerable value, we consider this component a weak spot in the current system, as the parser regularly produces incorrect analyses. In the current task, this was especially the case for the training data.

In the lexicalized syntactic features, as well as in the token n-grams, a token slot could also be filled with a word group indicator. We found candidates for these lists by searching the 2006 Google n-gram collection (Brants and Franz, 2006) with some regular expressions, and then cleaned them up manually. There were four word groups. `GrpADJgood` contains 100 positive adjectives, e.g. *better* and *advisable*, being the first 100 manually approved from 1774 sorted matches with e.g. `^it (would|could|might|may) be (.*) (to|if)`. `GrpADVinten` covers 37 intensifying adverbs, e.g. *very* and *critically*, selected from 448 matches with e.g. `^(would|could|might|may) be (.*) ($adjgood) (to|if)`. `GrpVadvise` contains 723 verbs that indicate what is being suggested, e.g. *adopt* and *edit*, selected from 987 matches with e.g. `^(i|we) (suggest|advise|propose|request) you to (.*)`. And `GrpADVdescr` has 41 adverbs that can modify the advised action, e.g. *always* and *never*, selected from 212 matches with e.g. `^(suggest|advise|propose|request) that you (.*) ($vadvise)`. When used in a feature, each group indicator is concatenated with the POS tag. The introduction of the word groups appears to be effective: one of the strongest markers, found 82 times with suggestions and only twice with non-suggestions, is "It would be GrpADJgoodJJ".

Our choice of recognition system was influenced by the desire to compare to the human rules. After successful recognition, we wanted to be able to identify which features contributed to the success. For this experiment, we chose a very simple algorithm. We counted the occurrences of each feature in the training items marked as suggestions or as non-suggestions and compared the two counts to derive odds. For example, the character 8-gram `Please a` was found 60 times in suggestion items and once in non-suggestion items. Correcting for the different numbers of items in the two classes, and adding one to avoid division by zero, this led to odds of 45.89 in favour of suggestions. In order to avoid exaggerated counts because of repeated items, we only used the first occurrence of each item, leaving us with 1758 suggestions and 5339 non-suggestions. This removal was done even if the repeated items had different suggestion labels. For the actual recognition, we only used features that had odds higher than or equal to 3:1. In the prediction phase, the odds of all features present in an item were taken and their sum was compared to a threshold, which we chose by tuning on the trial data.

As with other tasks, we investigated hybridization of our two approaches. In this case, straightforward combination of the final choices seemed not very fruitful on the trial data. However, when we compared the suggestions for each individual feature type in the machine learning with those of the expert rules, we found especially syntactic n-grams were able to identify suggestions not found by the rules, which were limited to contiguous n-grams. We therefore built a combination that marked those items as suggestions that were recognized as such by the rules, plus those for which the recognition score by only the syntactic n-gram features was over the optimal threshold for the trial data. The relative quality of rules and machine learning on the trial data for Subtask B made us decide not to attempt combination there.

4 Quantitative evaluation

In this discussion, we do not want to compare to other systems in the shared task. For this we refer the reader to the task description paper (Negi et al., 2019), where it can be seen that more intricate machine learning systems, especially those using pre-trained language models, perform much better. We will rather examine how our internally comparable systems behave on the four item sets. In order to make the measurements compatible, we first have to make some adjustments. Both trial sets contained equal numbers of suggestions and non-suggestions, so that precision and recall were equally valuable. In the evaluation sets, there were more non-suggestions than suggestions, so that precision has more influence than recall. For comparison we needed to recalculate precision (and hence F_1; recall is unchanged) by

Approach	Meas	TrialA	EvalA	EvalBalA	TrialB	EvalB	EvalBalB
Rules	Prec	0.8213	0.4521	0.8761	0.9673	0.8669	0.8991
	Rec	0.8851	0.7586	0.7586	0.8045	0.7299	0.7299
	F_1	0.8520	0.5665	0.8132	**0.8784**	**0.7925**	**0.8057**
Learn	Prec	0.7914	0.4848	0.8898	0.5802	0.5099	0.5873
	Rec	0.8716	0.7356	0.7356	0.9134	0.8851	0.8851
	F_1	0.8296	**0.5848**	0.8054	0.7096	0.6471	0.7061
Combo	Prec	0.8179	0.4558	0.8778		NA	
	Rec	0.8953	0.7701	0.7701		NA	
	F_1	**0.8548**	0.5726	**0.8204**		NA	
Baseline	Prec	0.5872	0.1566	0.6141	0.7277	0.6877	0.7507
	Rec	0.9324	0.9195	0.9195	0.8267	0.7845	0.7845
	F_1	0.7206	0.2676	0.7364	0.7740	0.7329	0.7672

Table 1: Quality of submitted systems and organiser baseline.

Approach	TrialA	EvalA	EvalBalA	TrialB	EvalB	EvalBalB
Weighted sum	**0.8291**	0.6429	**0.8110**	0.7109	0.6637	0.7170
Char n-grams	0.7968	**0.6444**	0.7781	0.6782	0.6113	0.6748
Token 1-gram	0.7492	0.5411	0.7444	0.6006	0.5320	0.5563
Token 2-gram	0.8045	0.5882	0.7100	0.6006	0.5256	0.5436
Token 3-gram	0.7968	0.5342	0.6436	0.3955	0.3363	0.3420
Token 4-gram	0.7664	0.4348	0.5023	0.1201	0.1027	0.1029
Structure main clause	0.7580	0.4667	0.5588	0.3866	0.3029	0.3090
Syntactic rewrites	0.4888	0.3068	0.4454	0.1849	0.2085	0.2126
Syntactic n-grams	0.7601	0.5069	0.7297	**0.7419**	**0.6946**	**0.7416**

Table 2: F_1-score for each test set for each feature type, using model learned from training material for Subtask A, and oracle thresholds.

extrapolating the systems' behaviour to a balanced testset. E.g., the expert rules had 80 false accepts on a total of 146, so a precision of $66/146$ i.e. 0.4521. In a balanced dataset, 87 instead of 746 non-suggestions, not 80/746, but 9.3298/87 would be falsely accepted. Precision would be $66/(66 + 9.3298)$ i.e. 0.8761. We stress that we are not trying to make out results look more impressive, but merely want to make behaviour on trial and evaluation sets comparable.

Table 1 shows the evaluation results for our three systems and the organisers' baseline, with the adjusted scores shown in the columns marked EvalBal. Here, we can see that the enormous drops in quality between TrialA and EvalA were an illusion, caused by the difference in balance. Also, our three systems are now consistent in their order: expert rules outperform machine learning, but the (very eclectic) combination scores yet a bit better. In fact, we now see that all systems gain precision when going from TrialA to EvalA, but at the cost of recall. For the machine learner, this is not exclusively due to overtraining in threshold optimization, as we see below in the discussion of Table 2. A discussion of the results for the expert rules can be found in Section 5. The lower degree of change for the baseline can probably be explained by the rather general level of the rules there. Going from Subtask A to Subtask B, the machine learner suffers a substantial loss in quality, which is understandable as it is trained only on Subtask A data. Interestingly, there are almost no differences between TrialB and EvalB. For the expert rules, the situation is rather different. From TrialA to TrialB, we see a precision gain and recall loss, leading to a slight increase in F_1; but the move from TrialB to EvalB leads to a serious drop in both precision and recall.

In Table 2, we show the F_1-score for each individual feature type, as well as for the sum for comparison. On TrialA, the most useful individual feature types appear to be token bi- and trigrams and

Feature	TrainA	TrialA	EvalA	TrialB	EvalB
Total items	1758/ 5339	296/ 296	87/ 746	404/ 404	348/ 476
`CG8_##Allow#`	63/2	14/0	5/0	0/0	0/0
`T4_WWWG_It_would_be_GrpADJgoodJJ`	82/2	13/1	4/0	0/0	0/0
`CG7_re#shou`	31/1	8/0	0/1*	0/0	0/0
`CG8_Please#a`	60/1	6/0	2/0	0/0	0/0
`SRFC_S_V_VP_A_VBx_OD_NP_A_PP`	28/0	4/0	1/0	0/0	1/0
`SCFFCCL_S_CS_AJP(nice)_A_SBAR(if)`	20/1	4/0	1/0	1/0	0/1
`CG8_#Provide`	20/0	3/0	1/0	0/0	0/0
`SCFFCCL_ROOT_UTT_S(suggest)_NOFUpunc_.(.)`	14/0	3/0	1/0	4/0	5/0
`T3_WWW_#_I_'d`	24/0	4/1	0/3*	2/0	1/0
`SCFFCCL_VP_AVB_MD(should)_MVB_VBN(GrpVadvise)`	46/5	4/1	3/1	1/0	3/1*

Table 3: Ten high-odds features (excluding correlated ones), with effectiveness on all test sets. Each cell contains observation counts in suggestions/non-suggestions. Hash marks indicate spaces and out-of-sentence positions. The asterisk (*) means we disagree on one of the non-suggestions, but did use the provided labels.

character n-grams. The other feature types lag behind, but do help reach a higher score with the sum of all feature scores. Syntactic rewrites by themselves have a much lower F_1, but this is due to their low recall potential (precision 0.7267, recall 0.3682). When moving to EvalA, we see that character n-grams and token unigrams maintain their quality, indicating that the same kind of words are being used, but higher n-grams and syntax lose severely, which suggests that EvalA is more different from the training data than TrialA in how words are combined.

When we proceed to Subtask B, all feature types lose quality. As we moved to a different domain, and a different genre, this is not surprising. Machine learning depends on having training and test data that is as similar as possible. It is encouraging to see that the syntactic n-grams do manage to perform similarly to Subtask A. This means that machine learning at a more abstract level is able to move to another domain more easily.

If we examine which features are responsible for the recognition, we see that all play some role. There are, however, some more effective ones. We show ten of these in Table 3. Note that this is a manual selection, as simply taking the ten highest scoring ones would show only two basic patterns in various guises, e.g. *Please* as several character n-grams, as token unigram, in a token bigram, and as an adverbial in a syntactic n-gram. Some of the ten patterns need explanation: `CG7_re#shou` is part of the token bigram *there should*. `SRFC_S_V_VP_A_VBx_OD_NP_A_PP` means that a sentence is being rewritten as a verb phrase (i.e. a sequence of verb with possibly additional internal adverbials), followed by an adverbial realized by a non-finite verb, then a direct object realized by a noun phrase, and finally an adverbial realized by a prepositional phrase. If we search for examples, we find e.g. *Please allow the access to phone filesystem.*. It turns out that Stanford CoreNLP marks *Please* as a verb, placing *allow* as head of an "xcomp" clause, which confuses our analysis transformer and makes *allow* an adverbial. This is clearly wrong but, as it is done consistently, this pattern still provides a good marker. `SCFFCCL_VP_AVB_MD(should)_MVB_VBN(GrpVadvise)` indicates that within a verb phrase, we find both the modal *should* and a past participle of one of the verbs for something that is advisable. `SCFFCCL_S_CS_AJP(nice)_A_SBAR(if)` represents a sentence with a subject complement *nice* and an adverbial clause headed by *if*. And `SCFFCCL_ROOT_UTT_S(suggest)_NOFUpunc_.(.)` represents a sentence with a main verb with lemma *suggest*, combined with a period as punctuation, which nicely rules out questions with *suggest*. All these patterns have a good precision, but their recall is obviously limited. The first two manage to get about 5 percent on Subtask A, then we quickly drop to 2.5 and 1. In general, recall is similar for TrialA and EvalA. However, the strongest markers are absent altogether in Subtask B, where suggestions are apparently worded differently. Only the final two syntactic n-grams show a significant presence, which is in line with the discussion on feature types above.

5 Qualitative evaluation

Upon inspection of the results obtained on the evaluation set with the expert rules, we specifically looked at the false accepts and false rejects. Contrary to what we thought might happen, only very few cases were missed out on due to omissions in the lexicon.

With cases that we missed (i.e. where we failed to recognize a suggestion) we did not find any clear clues as to what could be added to the patterns already specified in our rules set. There were some cases involving imperatives that we missed due to the fact that the punctuation mark we relied upon appeared to be absent, while gleaning the sentence type (interrogative) from the input final punctuation mark failed in cases where the input consisted of two or more sentences.

As for false accepts, we found that several cases were wrongly taken to be suggestions on the basis of a matching imperative pattern. Here the earlier problem of being unable to distinguish imperative verb forms from infinitives and present tenses no longer presented itself. Instead, word forms that are ambiguous between noun and verb such as *map* and *phone* were mistakenly held to be imperative verbs. Other false accepts were cases where otherwise highly successful rules proved to be too limited in their scope. For example, the pattern `AUXwould - Vinfin` would match *would allow* but then the sentence actually continues with *but* so that what initially looks like a suggestion turns out to be an observation, e.g. *The control would allow for the use of the contact picker but allows for manual entry and deletion of contacts*. Similar cases involving *but* were found with other patterns. In the cases of single instances slight modifications of the rules might have avoided wrongfully identifying something as a suggestion but there is very little evidence to go by, so there is no telling how it would impact on a different dataset.

We also noticed that some patterns occurred (almost) exclusively in Subtask A or B. Building separate rule sets for the two tasks might hence have been beneficial. However, the whole point of the exercise in Task 9 was to see whether a rule set can be ported to a new, mostly unknown, domain.

6 Discussion

The task was more difficult than we had expected. When we set out, we thought we knew what a suggestion was. However, after confronting our first version of the expert pattern system with the training data, we needed to review our ideas. After careful error analysis, we adapted our system and came to achieve quite acceptable results on the trial data, yet there remained a gap between what we (intuitively, as experienced language users) would consider a suggestion and quite a number of other cases which were labelled as such. In our view, there is a fine line between a suggestion and a non-suggestion, and perhaps one needs more context than a single sentence in order to tell which it is. We speculate that this may well explain the drop in performance. This is in line with the fact that the datasets provided, i.e. the training, trial and evaluation data, were rather different in nature: the training data comprised many inputs that were made up of multiple sentences, while the trial data and certainly the evaluation data mostly comprised single sentence inputs. Another factor which may have been at play here (and again we can only speculate), is that the annotations provided were made by different groups of annotators. This would then also explain some of the inconsistencies in the labeling, where similar cases were labeled differently.

Moving to our system, the expert-informed rule approach used previously with other tasks once more showed its strengths. The word n-gram patterns used are simple yet quite robust and effective, targeting specifically those parts of the input that are deemed relevant for the task at hand, without requiring these to be linguistically complete or well-formed phrases or clauses. There are no a priori limitations as to the length of the word n-grams. Compiling a lexicon and a set of rules requires limited effort. More than before, however, with the present task we experienced the limitations of using only contiguous word n-grams. Moreover, having only access to the raw text, information about the syntactic structure of a sentence was lacking, which in specific cases is key to being able to successfully identify a suggestion. We were able to fill some of this gap by combining with the ML approach, but improvements were as yet meagre as the parser was not performing optimally. Still, we have good hopes for this form of expert-informed hybridization.

References

Jan Aarts, Hans van Halteren, and Nelleke Oostdijk. 1998. The linguistic annotation of corpora: The TOSCA analysis system. *International journal of corpus linguistics*, 3(2):189–210.

Thorsten Brants and Alex Franz. 2006. *Web 1T 5-gram Version 1*. Linguistic Data Consortium, Philadelphia.

Hans van Halteren and Nelleke Oostdijk. 2018. Identification of differences between Dutch language varieties with the VarDial 2018 Dutch-Flemish subtitle data. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018), Santa Fe, New Mexico, USA*. Association for Computational Linguistics.

Ali Hürriyetoğlu, Nelleke Oostdijk, Mustafa Erkan Başar, and Antal van den Bosch. 2017. Supporting experts to handle tweet collections about significant events. In *International Conference on Applications of Natural Language to Information Systems*, pages 138–141. Springer.

Ali Hürriyetoğlu, Jurjen Wagemaker, Antal van den Bosch, and Nelleke Oostdijk. 2016. Analysing the role of key term inflections in knowledge discovery on twitter. In *Proceedings of the 2nd International Workshop on Knowledge Discovery on the Web (KDWEB16). Cagliari, Italy*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Nelleke Oostdijk, Ali Hürriyetoğlu, Marco Puts, Piet Daas, and Antal van den Bosch. 2016. Information extraction from the social media: A linguistically motivated approach. In *Actes de la conférence conjointe JEP-TALN-RECITAL 2016, volume 10: Risque et TAL: 21-33. PARIS Inalco du 4 au 8 juillet 2016*.

ThisIsCompetition at SemEval-2019 Task 9: BERT is unstable for out-of-domain samples

Cheoneum Park*[1] and **Juae Kim***[2] and **Hyeon-gu Lee***[1] and **Reinald Kim Amplayo***[3]
Harksoo Kim[1] and **Jungyun Seo**[2] and **Changki Lee**[1]
(* equal contribution)
[1]Kangwon National University, South Korea
[2]Sogang University, South Korea
[3]University of Edinburgh, UK

Abstract

This paper describes our system, Joint Encoders for Stable Suggestion Inference (**JESSI**), for the SemEval 2019 Task 9: Suggestion Mining from Online Reviews and Forums. JESSI is a combination of two sentence encoders: (a) one using multiple pre-trained word embeddings learned from log-bilinear regression (GloVe) and translation (CoVe) models, and (b) one on top of word encodings from a pre-trained deep bidirectional transformer (BERT). We include a domain adversarial training module when training for out-of-domain samples. Our experiments show that while BERT performs exceptionally well for in-domain samples, several runs of the model show that it is unstable for out-of-domain samples. The problem is mitigated tremendously by (1) combining BERT with a non-BERT encoder, and (2) using an RNN-based classifier on top of BERT. Our final models obtained second place with 77.78% F-Score on Subtask A (i.e. in-domain) and achieved an F-Score of 79.59% on Subtask B (i.e. out-of-domain), even without using any additional external data.

1 Introduction

Opinion mining (Pang and Lee, 2007) is a huge field that covers many NLP tasks ranging from sentiment analysis (Liu, 2012), aspect extraction (Mukherjee and Liu, 2012), and opinion summarization (Ku et al., 2006), among others. Despite the vast literature on opinion mining, the task on suggestion mining has given little attention. Suggestion mining (Brun and Hagège, 2013) is the task of collecting and categorizing suggestions about a certain product. This is important because while opinions indirectly give hints on how to improve a product (e.g. analyzing reviews), suggestions are direct improvement requests (e.g. tips, advice, recommendations) from people who have used the product.

To this end, Negi et al. (2019) organized a shared task specifically on suggestion mining called SemEval 2019 Task 9: Suggestion Mining from Online Reviews and Forums. The shared task is composed of two subtasks, Subtask A and B. In Subtask A, systems are tasked to predict whether a sentence of a certain domain (i.e. electronics) entails a suggestion or not given a training data of the same domain. In Subtask B, systems are tasked to do suggestion prediction of a sentence from another domain (i.e. hotels). Organizers observed four main challenges: (a) sparse occurrences of suggestions; (b) figurative expressions; (c) different domains; and (d) complex sentences. While previous attempts (Ramanand et al., 2010; Brun and Hagège, 2013; Negi and Buitelaar, 2015) made use of human-engineered features to solve this problem, the goal of the shared task is to leverage the advancements seen on neural networks, by providing a larger dataset to be used on data-intensive models to achieve better performance.

This paper describes our system **JESSI** (Joint Encoders for Stable Suggestion Inference). JESSI is built as a combination of two neural-based encoders using multiple pre-trained word embeddings, including BERT (Devlin et al., 2018), a pre-trained deep bidirectional transformer that is recently reported to perform exceptionally well across several tasks. The main intuition behind JESSI comes from our finding that although BERT gives exceptional performance gains when applied to in-domain samples, it becomes unstable when applied to out-of-domain samples, even when using a domain adversarial training (Ganin et al., 2016) module. This problem is mitigated using two tricks: (1) jointly training BERT with a CNN-based encoder, and (2) using an RNN-based encoder on top of BERT before feeding to the classifier.

JESSI is trained using only the datasets given on the shared task, without using any additional external data. Despite this, JESSI performs second on Subtask A with an F1 score of 77.78% among 33 other team submissions. It also performs well on Subtask B with an F1 score of 79.59%.

2 Related Work

Suggestion Mining The task of detecting suggestions in sentences is a relatively new task, first mentioned in Ramanand et al. (2010) and formally defined in Negi and Buitelaar (2015). Early systems used manually engineered patterns (Ramanand et al., 2010) and rules (Brun and Hagège, 2013), and linguistically motivated features (Negi and Buitelaar, 2015) trained on a supervised classifier (Negi et al., 2016). Automatic mining of suggestion has also been suggested (Dong et al., 2013). Despite the recent successes of neural-based models, only few attempts were done, by using neural network classifiers such as CNNs and LSTMs (Negi et al., 2016), by using part-of-speech embeddings to induce distant supervision (Negi and Buitelaar, 2017). Since neural networks are data hungry models, a large dataset is necessary to optimize the parameters. SemEval 2019 Task 9 (Negi et al., 2019) enables training of deeper neural models by providing a much larger training dataset.

Domain Adaptation In text classification, training and test data distributions can be different, and thus domain adaptation techniques are used. These include non-neural methods that map the semantics between domains by aligning the vocabulary (Basili et al., 2009; Pan et al., 2010) and generating labeled samples (Wan, 2009; Yu and Jiang, 2016). Neural methods include the use of stacked denoising autoencoders (Glorot et al., 2011), variational autoencoders (Saito et al., 2017; Ruder and Plank, 2018). Our model uses a domain adversarial training module (Ganin et al., 2016), an elegant way to effectively transfer knowledge between domains by training a separate domain classifier using an adversarial objective.

Language Model Pretraining Inspired from the computer vision field, where ImageNet (Deng et al., 2009) is used to pretrain models for other tasks (Huh et al., 2016), many recent attempts in the NLP community are successful on using language modeling as a pretraining step to extract

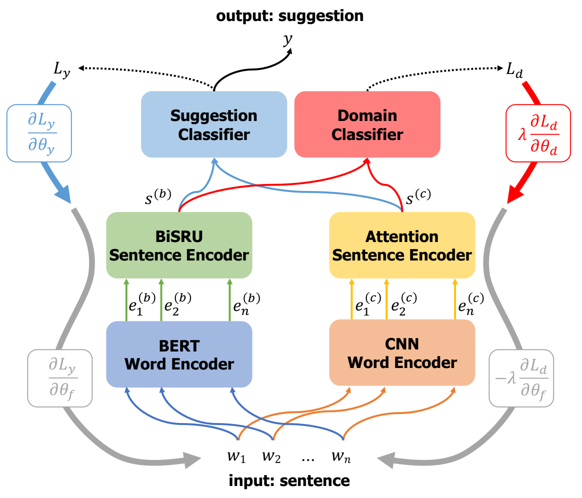

output: suggestion
y
L_y
L_d
Suggestion Classifier
Domain Classifier
$\frac{\partial L_y}{\partial \theta_y}$
$\lambda \frac{\partial L_d}{\partial \theta_d}$
$s^{(b)}$
$s^{(c)}$
BiSRU Sentence Encoder
Attention Sentence Encoder
$e_1^{(b)}$ $e_2^{(b)}$ $e_n^{(b)}$
$e_1^{(c)}$ $e_2^{(c)}$ $e_n^{(c)}$
BERT Word Encoder
CNN Word Encoder
$\frac{\partial L_y}{\partial \theta_f}$
$-\lambda \frac{\partial L_d}{\partial \theta_f}$
w_1 w_2 ... w_n
input: sentence

Figure 1: The overall architecture of JESSI for Subtask B. The thinner arrows correspond to the forward propagations, while the thicker arrows correspond to the backward propagations, where gradient calculations are indicated. For Subtask A, a CNN encoder is used instead of the BiSRU encoder, and the domain adversarial training module is not used.

feature representations (Peters et al., 2018), and to fine-tune NLP models (Radford et al., 2018; Devlin et al., 2018). BERT (Devlin et al., 2018) is the most recent inclusion to these models, where it uses a deep bidirectional transformer trained on masked language modeling and next sentence prediction objectives. Devlin et al. (2018) reported that BERT shows significant increase in improvements on many NLP tasks, and subsequent studies have shown that BERT is also effective on harder tasks such as open-domain question answering (Yang et al., 2019), multiple relation extraction (Wang et al., 2019), and table question answering (Hwang et al., 2019), among others. In this paper, we also use BERT as an encoder, show its problem on out-of-domain samples, and mitigate the problem using multiple tricks.

3 Joint Encoders for Stable Suggestion Inference

We present our model **JESSI**, which stands for Joint Encoders for Stable Suggestion Inference, shown in Figure 1. Given a sentence $x = \{w_1, w_2, ..., w_n\}$, JESSI returns a binary suggestion label $y = \{0, 1\}$. JESSI consists of four important components: (1) A BERT-based encoder that leverages general knowledge acquired from a large pre-trained language model, (2) A CNN-based encoder that learns task-specific sentence representations, (3) an MLP classifier that predicts the label given the joint encodings, and (4) a domain adversarial training module that prevents the model to distinguish between the two domains.

BERT-based Encoder Fine-tuning a pre-trained BERT (Devlin et al., 2018) classifier then using the separately produced classification encoding [CLS] has shown to produce significant improvements. Differently, JESSI uses a pre-trained BERT as a word encoder, that is instead of using [CLS], we use the word encodings $e_1^{(b)}, e_2^{(b)}, ..., e_n^{(b)}$ produced by BERT. BERT is still fine-tuned during training.

We append a sentence encoder on top of BERT, that returns a sentence representation $s^{(b)}$, which is different per subtask. For Subtask A, we use a CNN encoder with max pooling (Kim, 2014) to create the sentence embedding. For Subtask B, we use a bidirectional simple recurrent units (Lei et al., 2018, BiSRU), a type of RNN that is highly parallelizable, as the sentence encoder.

CNN-based Encoder To make the final classifier more task-specific, we use a CNN-based encoder that is trained from scratch. Specifically, we employ a concatenation of both pre-trained GloVe (Pennington et al., 2014) and CoVe (McCann et al., 2017) word embeddings as input $w_i, 1 \leq i \leq n$. Then, we do convolution operations $\text{Conv}(w_i, h_j)$ using multiple filter sizes h_j to a window of h_j words. We use different paddings for different filter sizes such that the number of output for each convolution operation is n. Finally, we concatenate the outputs to obtain the word encodings, i.e. $e_i^{(c)} = \oplus_j(\text{Conv}(w_i, h_j))$, where $\oplus$ is the sequence concatenate operation.

We pool the word encodings using attention mechanism to create a sentence representation $s^{(c)}$. That is, we calculate attention weights using a latent variable v that measures the importance of the words $e_i^{(c)}$, i.e., $a_i = \text{softmax}(v^\top f(e_i^{(c)}))$, where $f(\cdot)$ is a nonlinear function. We then use a_i to weight-sum the words into one encoding, i.e., $s^{(c)} = \sum_i a_i e_i^{(c)}$.

Suggestion Classifier Finally, we use a multi-layer perceptron (MLP) as our classifier, using a concatenation of outputs from both the BERT- and CNN-based encoders, i.e., $p(y) = \text{MLP}_y([s^{(b)}; s^{(c)}])$. Training is done by minimizing the cross entropy loss, i.e., $\mathbb{L} = -\log p(y)$.

Domain Adversarial Training For Subtask B, the model needs to be able to classify out-of-domain samples. Using the model as is decreases performance significantly because of cross-domain differences. To this end, we use a domain adversarial training module (Ganin et al., 2016) to prevent the classifier on distinguishing differences between domains. Specifically, we create another MLP classifier that classifies the *domain* of the text using the concatenated sentence encoding with *reverse gradient function* $\text{GradRev}(\cdot)$, i.e., $p(d) = \text{MLP}_d(\text{GradRev}([s^{(b)}; s^{(c)}]))$. The reverse gradient function is a function that performs equivalently with the identity function when propagating forward, but reverses the sign of the gradient when propagating backward.

Through this, we eliminate the possible ability of the classifier to distinguish the domains of the text. We train the domain classifier using the available trial datasets for each domain. We also use a cross entropy loss as the objective of this classifier. Overall, the objective of JESSI is to minimize the following loss: $\mathbb{L} = -\log p(y) - \lambda \log p(d)$, where λ is set increasingly after each epoch, following Ganin et al. (2016).

4 Experimental Setup

Dataset and Preprocessing We use the dataset provided in the shared task: a training dataset from the electronics domain, and labeled trial and unlabeled test datasets from both the electronics and hotels domain. Table 1 summarizes the dataset statistics and shows the distribution differences between two domains. During training, we use the labeled training dataset to train the suggestions classifier, and trial datasets, without the suggestion labels, to train the domains classifier. For preprocessing, we lowercased and tokenized using the Stanford CoreNLP toolkit[1] (Manning et al., 2014).

Implementation We use the pre-trained BERT models[2] provided by the original authors to initialize the parameters of BERT. We use BERT-large

[1]https://stanfordnlp.github.io/CoreNLP/

[2]https://github.com/google-research/bert

Subtask	A	B
Domain	Electronics	Hotels
#Training	8,230	0
#Trial	592	808
#Test	833	824
#Vocabulary	10,897	3,570
Ave. Tokens	19.0	16.8

Table 1: Dataset Statistics

for Subtask A and BERT-base for Subtask B[3]. For our CNNs, we use three filters with sizes $\{3, 5, 7\}$, each with 200 dimensions. For the BiSRU, we use hidden states with 150 dimensions and stack with two layers. The MLP classifier contains two hidden layers with 300 dimensions.

We use dropout (Srivastava et al., 2014) on all nonlinear connections with a dropout rate of 0.5. We also use an l_2 constraint of 3. During training, we use mini-batch size of 32. Training is done via stochastic gradient descent over shuffled mini-batches with the Adadelta (Zeiler, 2012) update rule. We perform early stopping using the trial sets. Moreover, since the training set is relatively small, multiple runs lead to different results. To handle this, we perform an ensembling method as follows. We first run 10-fold validation over the training data, resulting into ten different models. We then pick the top three models with the highest performances, and pick the class with the most model predictions.

5 Experiments

In this section, we show our results and experiments. We denote JESSI-A as our model for Subtask A (i.e., BERT→CNN+CNN→ATT), and JESSI-B as our model for Subtask B (i.e., BERT→BISRU+CNN→ATT+DOMADV). The performance of the models is measured and compared using the F1-score.

Ablation Studies We present in Table 2 ablations on our models. Specifically, we compare JESSI-A with the same model, but without the CNN-based encoder, without the BERT-based encoder, and with the CNN sentence encoder of the BERT-based encoder replaced with the BiSRU variant. We also compare JESSI-B with the same model, but without the CNN-based encoder, without the BERT-based encoder, without the domain adversarial training module, and with the BiSRU sentence encoder of the BERT-based encoder replaced with the CNN variant. The ablation studies show several observations. First, jointly combining both BERT- and CNN-based encoders help improve the performance on both subtasks. Second, the more effective sentence encoder for the BERT-based encoder (i.e., CNN versus BiSRU) differs for each subtask; the CNN variant is better for Subtask A, while the BiSRU variant is better for Subtask B. Finally, the domain adversarial training module is very crucial in achieving a significant increase in performance.

[3]Due to memory limitations, we are limited to use the smaller BERT model for Subtask B. We expect an increase in performance when BERT-large is used.

Model	F-Score
JESSI-A	88.78
+ BERT→BISRU	86.01
– CNN→ATT	85.14
– BERT→CNN	83.89

(a) Subtask A

Model	F-Score
JESSI-B	87.31
– CNN→ATT	84.01
– BERT→BISRU	81.13
+ BERT→CNN	70.21
– DOMADV	47.48

(b) Subtask B

Table 2: Ablation results for both subtasks using the provided trial sets. The + denotes a *replacement* of the BERT-based encoder, while the – denotes a *removal* of a specific component.

Out-of-Domain Performance During our experiments, we noticed that BERT is unstable when predicting out-of-domain samples, even when using the domain adversarial training module. We show in Table 3 the summary statistics of the F-Scores of 10 runs of the following models: (a) vanilla BERT that uses the [CLS] classification encoding, (b-c) our BERT-based encoders BERT→CNN and BERT→BISRU that use BERT as a word encoder and use an additional CNN/BiSRU as a sentence encoder, (d) JESSI-B that uses BERT→BISRU and CNN→ATT as joint encoders, and (e) CNN→ATT that does not employ BERT in any way. The results show that while CNN→ATT performs similarly on different runs, BERT performs very unsta-

Model	min	max	mean	std
BERT	0.00	70.59	22.52	31.0
BERT→CNN	0.00	74.62	28.23	34.1
BERT→BiSRU	54.00	88.83	74.86	8.8
JESSI-B	69.28	89.21	82.41	5.6
CNN→Att	68.19	77.06	72.50	2.5

Table 3: Summary statistics of the F-Scores of 10 runs of different models on the trial set of Subtask B when doing a 10-fold validation over the available training data. All models include the domain adversarial training module (+DomAdv), which is omitted for brevity.

bly, achieving varying F-Scores as low as zero and as high as 70.59, with a standard deviation of 31. Appending a CNN-based sentence encoder (i.e., BERT→CNN) increases the performance, but worsens the stability of the model. Appending an RNN-based sentence encoder (i.e., BERT→BiSRU) both increases the performance and improves the model stability. Finally, combining a separate CNN-based encoder (i.e., JESSI-B) improves the performance and stability further.

Test Set Results Table 4 presents how JESSI compared to the top performing models during the competition proper. Overall, JESSI-A ranks second out of 33 official submissions with an F-Score of 77.78%. Although we were not able to submit JESSI-B during the submission phase, JESSI-B achieves an F-Score of 79.59% on the official test set. This performance is similar to the performance of the model that obtained sixth place in the competition. We emphasize that JESSI does not use any labeled and external data for Subtask B, and thus is just exposed to the hotels domain using the available *unlabeled* trial dataset, containing 808 data instances. We expect the model to perform better when additional data from the hotels domain.

Performance by Length We compare the performance of models on data with varying lengths to further investigate the increase in performance of JESSI over other models. More specifically, for each range of sentence length (e.g., from 10 to 20), we look at the accuracy of JESSI-A, BERT→BiSRU, and BERT→CNN on Subtask A, and the accuracy of JESSI-B, BERT→BiSRU, and BERT→CNN, all with domain adversarial training module, on Subtask B. Figure 2 shows the plots of the experiments on both subtasks. On both experiments, JESSI outperforms the other models

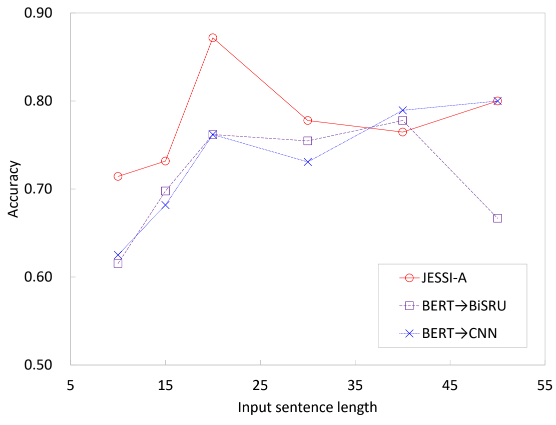
Rank	Model	F-Score
1	OleNet	78.12
2	JESSI-A	77.78
3	m_y	77.61

(a) Subtask A

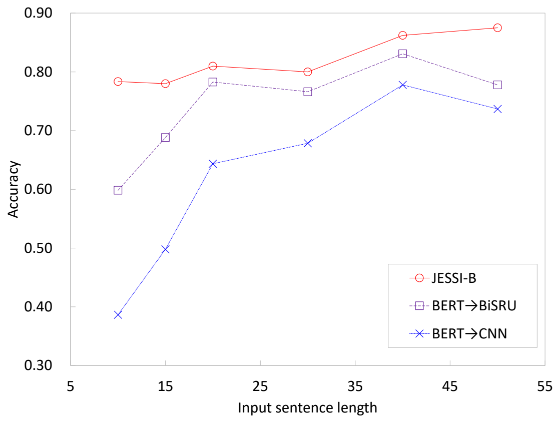
Rank	Model	F-Score
1	NTUA-ISLab	85.80
2	OleNet	85.79
3	NL-FIIT	83.13
*	JESSI-B	79.59
11	CNN→Att+DomAdv	64.86

(b) Subtask B

Table 4: F-Scores of JESSI and top three models for each subtask. Due to time constraints, we were not able to submit JESSI-B during the competition. For clarity, we also show our final official submission (CNN→Att+DomAdv).



(a) Subtask A



(b) Subtask B

Figure 2: Accuracy over various input sentence length on the test set.

when the sentence length is short, suggesting that the increase in performance of JESSI can be attributed to its performance in short sentences. This is more evident in Subtask B, where the difference of accuracy between JESSI and the next best model is approximately 20%. We can also see a consistent increase in performance of JESSI over other models on Subtask B, which shows the robustness of JESSI when predicting out-of-domain samples.

6 Conclusion

We presented JESSI (Joint Encoders for Stable Suggestion Inference), our system for the SemEval 2019 Task 9: Suggestion Mining from Online Reviews and Forums. JESSI builds upon jointly combined encoders, borrowing pre-trained knowledge from a language model BERT and a translation model CoVe. We found that BERT alone performs bad and unstably when tested on out-of-domain samples. We mitigate the problem by appending an RNN-based sentence encoder above BERT, and jointly combining a CNN-based encoder. Results from the shared task show that JESSI performs competitively among participating models, obtaining second place on Subtask A with an F-Score of 77.78%. It also performs well on Subtask B, with an F-Score of 79.59%, even without using any additional external data.

Acknowledgement

This research was supported by the MSIT (Ministry of Science ICT), Korea, under (National Program for Excellence in SW) (2015-0-00910) and (Artificial Intelligence Contact Center Solution) (2018-0-00605) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation)

References

Roberto Basili, Diego De Cao, Danilo Croce, Bonaventura Coppola, and Alessandro Moschitti. 2009. Cross-language frame semantics transfer in bilingual corpora. In *Computational Linguistics and Intelligent Text Processing, 10th International Conference, CICLing 2009, Mexico City, Mexico, March 1-7, 2009. Proceedings*, pages 332–345.

Caroline Brun and Caroline Hagège. 2013. Suggestion mining: Detecting suggestions for improvement in users' comments. *Research in Computing Science*, 70:199–209.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Li Dong, Furu Wei, Yajuan Duan, Xiaohua Liu, Ming Zhou, and Ke Xu. 2013. The automated acquisition of suggestions from tweets. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA.*

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor S. Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17:59:1–59:35.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 513–520.

Mi-Young Huh, Pulkit Agrawal, and Alexei A. Efros. 2016. What makes imagenet good for transfer learning? *CoRR*, abs/1608.08614.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Lun-Wei Ku, Yu-Ting Liang, and Hsin-Hsi Chen. 2006. Opinion extraction, summarization and tracking in news and blog corpora. In *Computational Approaches to Analyzing Weblogs, Papers from the 2006 AAAI Spring Symposium, Technical Report SS-06-03, Stanford, California, USA, March 27-29, 2006*, pages 100–107.

Tao Lei, Yu Zhang, Sida I. Wang, Hui Dai, and Yoav Artzi. 2018. Simple recurrent units for highly parallelizable recurrence. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4470–4481.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, System Demonstrations*, pages 55–60.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6297–6308.

Arjun Mukherjee and Bing Liu. 2012. Aspect extraction through semi-supervised modeling. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*, pages 339–348.

Sapna Negi, Kartik Asooja, Shubham Mehrotra, and Paul Buitelaar. 2016. A study of suggestions in opinionated texts and their automatic detection. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics, *SEM@ACL 2016, Berlin, Germany, 11-12 August 2016*.

Sapna Negi and Paul Buitelaar. 2015. Towards the extraction of customer-to-customer suggestions from reviews. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 2159–2167.

Sapna Negi and Paul Buitelaar. 2017. Inducing distant supervision in suggestion mining through part-of-speech embeddings. *CoRR*, abs/1709.07403.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 751–760.

Bo Pang and Lillian Lee. 2007. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237.

Alec Radford, Karthik Narasimhan, Time Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, Technical report, OpenAI.

J. Ramanand, Krishna Bhavsar, and Niranjan Pedanekar. 2010. Wishful thinking - finding suggestions and 'buy' wishes from product reviews. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 54–61. Association for Computational Linguistics.

Sebastian Ruder and Barbara Plank. 2018. Strong baselines for neural semi-supervised learning under domain shift. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1044–1054.

Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. 2017. Asymmetric tri-training for unsupervised domain adaptation. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 2988–2997.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 235–243.

Haoyu Wang, Ming Tan, Mo Yu, Shiyu Chang, Dakuo Wang, Kun Xu, Xiaoxiao Guo, and Saloni Potdar. 2019. Extracting multiple-relations in one-pass with pre-trained transformers. *arXiv preprint arXiv:1902.01030*.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718*.

Jianfei Yu and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *Proceedings of the 2016*

Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016, pages 236–246.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

WUT at SemEval-2019 Task 9: Domain-Adversarial Neural Networks for Domain Adaptation in Suggestion Mining

Mateusz Klimaszewski **Piotr Andruszkiewicz**
Institute of Computer Science
Warsaw University of Technology, Poland
mk.klimaszewski@gmail.com, P.Andruszkiewicz@ii.pw.edu.pl

Abstract

We present a system for cross-domain suggestion mining, prepared for the SemEval-2019 Task 9: Suggestion Mining from Online Reviews and Forums (Subtask B). Our submitted solution for this text classification problem explores the idea of treating different suggestions' sources as one of the settings of Transfer Learning - Domain Adaptation. Our experiments show that without any labeled target domain examples during training time, we are capable of proposing a system, reaching up to 0.778 in terms of F_1 score on test dataset, based on Target Preserving Domain-Adversarial Neural Networks.

1 Introduction

Suggestion mining is an emerging task in a natural language processing (NLP) field. Definition of suggestion mining task differs in NLP's community. Close areas of study like opinion mining or sentiment analysis get a lot of attention not only from academic, but also industrial researchers. From a linguistic point of view, while these areas treat neutral polarity of a statement as an absence of opinion (Liu, 2009), suggestion does not have to be connected with positive or negative emotion and can be treated as complementary information (Negi and Buitelaar, 2015). Lack of sensitivity to statement's sentiment and various suggestions' realization strategies (Martínez Flor, 2005) make suggestion mining task interesting and challenging from a NLP's standpoint.

In this work, we present a system for cross-domain suggestion mining, ranked in the 7^{th} place in SemEval-2019 Task 9 Subtask B. The training data for this task was collected from feedback posts on Universal Windows Platform. On the other hand, the test dataset comes from the different domain of hotel reviews from the TripAdvisor website (Negi et al., 2019). In this work we will refer to those datasets' domain as *source domain* and *target domain* accordingly. For suggestion mining task in this context, we employ ensemble of Domain-Adversarial Neural Networks (DANN) where we use Structured Self-Attentive Sentence Embedding (Lin et al., 2017) as a feature extractor. Moreover, to achieve better adaptation towards target domain, we follow the approach of Gui et al. (2017) for the part-of-speech tagging and extend DANN with a Target Preserving component in a form of words decoder for target domain sentences. We train all of the parts of the described system using modified domain adversarial training procedure than the one proposed in (Ganin et al., 2016).

2 Data preparation

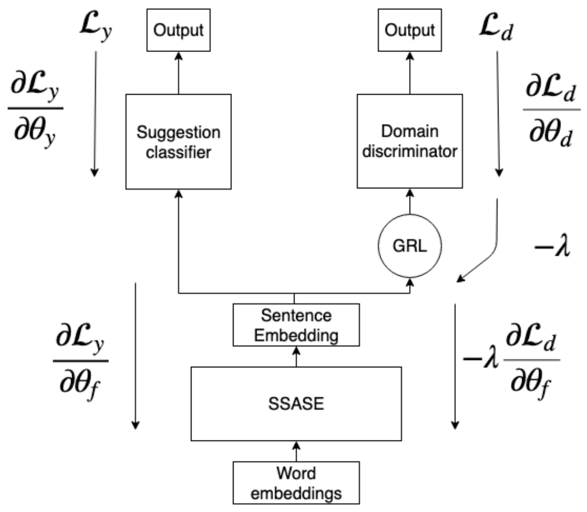
2.1 Dataset augmentation

Training dataset for Subtask B was built using only sentences from source domain. In order to train DANN we take advantage of an additional set of unlabeled sentences from the same domain as a test dataset. We use a subset of data from another corpora (Wachsmuth et al., 2014) consisting of hotel's reviews.

The selection of the subset is as follows, first we take benefit of a "weak" classifier in the form of the baseline, rules-based system provided by the organizers, to predict a class in the mentioned corpora. After that we choose the subset with the same distribution of classes (2085 *suggestions* and 6415 *no suggestions*) and remove too short statements to obtain a histogram of sentences' lengths close to the rest of datasets.

2.2 Preprocessing

We use Keras (Chollet et al., 2015) to perform preprocessing such as removing punctuation signs and lower-casing sentences. Considering that our



$\mathcal{L}_y$
Output
Output
$\mathcal{L}_d$
$\frac{\partial \mathcal{L}_y}{\partial \theta_y}$
Suggestion classifier
Domain discriminator
$\frac{\partial \mathcal{L}_d}{\partial \theta_d}$
GRL
$-\lambda$
Sentence Embedding
$\frac{\partial \mathcal{L}_y}{\partial \theta_f}$
SSASE
$-\lambda \frac{\partial \mathcal{L}_d}{\partial \theta_f}$
Word embeddings

Figure 1: DANN for cross-domain suggestion mining task.

model is based on recurrent neural network, we also pad sentences or shorten them to have maximum count of 50 tokens. Finally, as the source domain has a lot of urls, we replace all of them with a single *"https"* token.

2.3 Word embeddings

Last step is mapping sentences to mathematically computable form. We leverage the existing language models, which are pre-trained on huge volumes of raw text. Following the recent research, showing superiority of the contextual word embeddings over theirs predecessors, we apply Embeddings from Language Models (ELMo) (Peters et al., 2018) provided by Tensorflow Hub (Abadi et al., 2015). However, we do not fine-tune them with our model.

3 Model description

3.1 Domain-Adversarial Neural Networks

Ganin et al. (2015; 2016) proposed a system for unsupervised Domain Adaptation problem, adaptable to any neural network architecture. It consists of three components: a feature extractor, a label predictor and a domain discriminator. Last one, thanks to gradient reversal layer (GRL), which reverses flow of a gradient with respect to hyperparameter λ, allows to force the feature extractor to learn domain-invariant representations. Fig. 1 presents high level overview of the proposed architecture.

DANN minimizes loss presented in Eq. 1, where y stands for suggestion classifier, d domain descriptor and f for feature extractor presented in the following Section. An upper index in loss $\mathcal{L}$ symbolizes examples' domain.

$$E(\theta_f,\theta_y,\theta_d) = \frac{1}{n}\sum_{s=1}^{n}\mathcal{L}_y^s(\theta_f,\theta_y) - \lambda(\frac{1}{n}\sum_{s=1}^{n}\mathcal{L}_d^s(\theta_f,\theta_d) + \frac{1}{n'}\sum_{t=n+1}^{N}\mathcal{L}_d^t(\theta_f,\theta_d)) \tag{1}$$

3.2 Structured Self-Attentive Sentence Embedding

We model sentences using Structured Self-Attentive Sentence Embedding (SSASE) as feature extractor. Taking ELMo word embeddings as an input, followed by Bidirectional LSTM (BiLSTM) layer, SSASE used extended self-attention represented as an attention matrix (A) regularized by a penalization term as in Eq. 2, where $||\bullet||_F$ is Frobenius norm of a matrix and I is an identity matrix. Impact of penalization is controlled by hyperparameter α.

$$P = \alpha||(AA^T - I)||_F^2 \tag{2}$$

The attention matrix is calculated as shown in Eq. 3, where H is BiLSTM concatenated output, W_1 and W_2 are matrices of weights.

$$A = softmax(W_2 tanh(W_1 H^T)) \tag{3}$$

3.3 Target Preserving component

To prevent erasing targets domain specific features, we extend DANN with a target domain words decoder (model further referred as TPDANN-SSASE). The decoder is formed by an LSTM layer followed by one fully-connected layer. It takes as input a matching timestep of SSASE's BiLSTM outputs and predicts the input word.

In terms of the objective function, decoder's loss was limited by hyperparameter $\gamma = 0.4$ as in Eq. 4, where θ_r stands for decoder's parameters and θ_f^* - parameters of feature extractor's BiLSTM.

$$E(\theta_f,\theta_y,\theta_d,\theta_r) = E(\theta_f,\theta_y,\theta_d) + \gamma\frac{1}{n'}\sum_{t=n+1}^{N}\mathcal{L}_r^t(\theta_f^*,\theta_r) \tag{4}$$

3.4 Training algorithm

We apply a modification of domain-adversarial training procedure (Ganin et al., 2016). We treat an architecture as two separate networks with shared parameters of a feature extractor and a domain descriptor (θ_d and θ_f). In each training step, taking regular DANN as an example, we first update parameters trained using source domain Eq. 5, 6, 7 and then with a target domain examples as shown in Eq. 8, 9, where θ' stands for temporal state of parameters between those two updates and η denotes learning rate. The proposed change in the learning algorithm has been beneficial in terms of exploration properties.

$$\theta'_f \longleftarrow \theta_f - \eta(\frac{\partial \mathcal{L}^s_y}{\partial \theta_f} - \lambda \frac{\partial \mathcal{L}^s_d}{\partial \theta_f}) \tag{5}$$

$$\theta_y \longleftarrow \theta_y - \eta \frac{\partial \mathcal{L}^s_y}{\partial \theta_y} \tag{6}$$

$$\theta'_d \longleftarrow \theta_d - \eta \frac{\partial \mathcal{L}^s_d}{\partial \theta_d} \tag{7}$$

$$\theta_d \longleftarrow \theta'_d - \eta \frac{\partial \mathcal{L}^t_d}{\partial \theta'_d} \tag{8}$$

$$\theta_f \longleftarrow \theta'_f - \eta(-\lambda \frac{\partial \mathcal{L}^t_d}{\partial \theta'_f}) \tag{9}$$

4 Evaluation

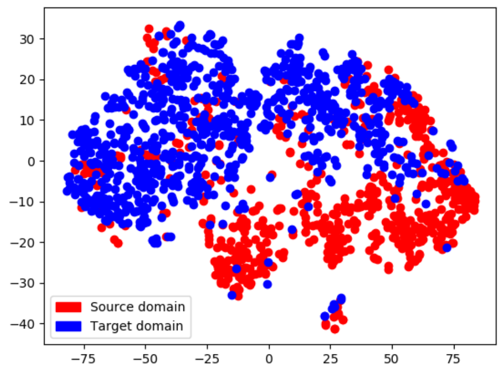
4.1 Results

The metric which was taken into account in SemEval-2019 Task 9 Subtask B was F_1 score. Table 1 presents results for tested architectures for validation and test datasets. Our baseline method is *fastText* (Joulin et al., 2017). It achieves higher score ($F_1 = 0.684$) on Subtask's A validation dataset (source domain) than on target domain, indicating that there is a shift between domains. We notice the same behaviour while testing SSASE model with only a label classifier. By adding domain adaptation components we manage to limit that problem. DANN-SSASE* is trained using default domain-adversarial training procedure (Ganin et al., 2016), while further models benefit from our proposed algorithm. We achieve final score, resulting in the 7^{th} place, by creating unweighted ensemble of three TPDANN-SSASE models.

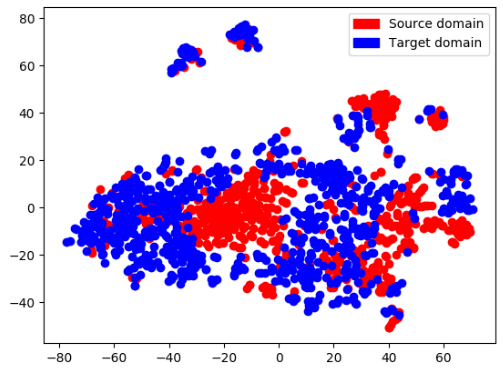
4.2 Hyperparameters

We use default ELMo embeddings with length of 1024. Each LSTM layer has 300 units (BiLSTM 600). Attention matrix dimensions are accordingly equal to 400 and 9 for W_1 and W_2. We set a penalization hyperparameter α to 0.45.

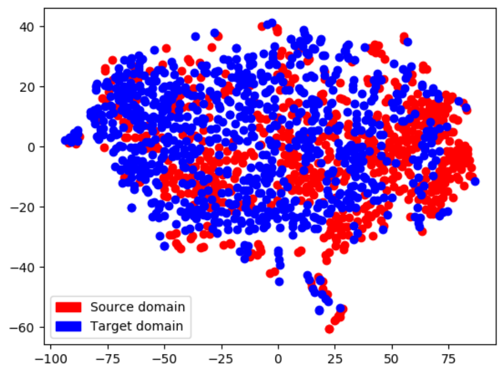
4.3 Domains shift



(a) SSASE



(b) DANN-SSASE



(c) TPDANN-SSASE

Figure 2: Domains shift reduction between models.

Method	Validation dataset	Test dataset
fastText	0.532	0.591
SSASE	0.517	0.467
DANN-SSASE*	0.616	0.558
DANN-SSASE	0.781	0.753
TPDANN-SSASE	0.831	0.764
TPDANN-SSASE ensemble	0.836	0.778

Table 1: F_1 score on target domain validation and test datasets.

Method	Source	Target
SSASE	8.26	8.78
DANN-SSASE	8.08	8.55
TPDANN-SSASE	7.24	7.56

Table 2: Mean count of the 10 nearest neighbours from the same domain. Desired score is equal to 5. To build kNN model, a representation of sentences was extracted from the last layer of a feature extractor and distance was measured using a Euclidean distance.

To measure a problem of domains shift and impact of domain adaptation components in our models we propose a metric based on number of nearest neighbours from the same domain. Assuming that there is no shift between domains, mean number of k nearest neighbours from particular domain over the whole dataset is equal to $\frac{k}{2}$. On the other hand to perfect overlap, it would be equal to k, as each sample could only have neighbours from the same domain.

We take the last layer of a feature extractor as the representations for which euclidean distance metric was employed to find nearest neighbours. Results presented in Tab. 2 indicate that models with better domain-invariant properties have better results in terms of suggestion mining task, TPDANN-SSASE achieves the closest values to $\frac{k}{2}$. In order to present how the domains overlap changed over models, we visualize them using T-SNE (van der Maaten and Hinton, 2008) (Fig. 2). The visualization confirms results presented in Tab. 2 - we observe the highest overlap for TPDANN-SSASE.

5 Conclusion

In this work, we introduced a new system for cross-domain suggestion mining based on the domain-adversarial neural networks. Domains shift reduction led to improvement of classification accuracy in target domain. Our proposed modification of adversarial training procedure allowed ensemble of TPDANN-SASSE models to reach F_1 value of 0.778.

References

Martín Abadi et al. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

François Chollet et al. 2015. Keras. `https://keras.io`.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1180–1189, Lille, France. PMLR.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35.

Tao Gui, Qi Zhang, Haoran Huang, Minlong Peng, and Xuanjing Huang. 2017. Part-of-speech tagging for twitter with adversarial neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2411–2420. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association* *for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding.

Bing Liu. 2009. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Springer-Verlag, Berlin, Heidelberg.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

Alicia Martínez Flor. 2005. A theoretical review of the speech act of suggesting: towards a taxonomy for its use in flt. *Revista Alicantina de Estudios Ingleses*.

Sapna Negi and Paul Buitelaar. 2015. Towards the extraction of customer-to-customer suggestions from reviews. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2159–2167, Lisbon, Portugal. Association for Computational Linguistics.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, Gregor Engels, and Tsvetomira Palakarska. 2014. A review corpus for argumentation analysis. In *Computational Linguistics and Intelligent Text Processing*, pages 115–127, Berlin, Heidelberg. Springer Berlin Heidelberg.

Yimmon at SemEval-2019 Task 9: Suggestion Mining with Hybrid Augmented Approaches

Yimeng Zhuang
Samsung Research China - Beijing (SRC-B)
ym.zhuang@samsung.com

Abstract

Suggestion mining task aims to extract tips, advice, and recommendations from unstructured text. The task includes many challenges, such as class imbalance, figurative expressions, context dependency, and long and complex sentences. This paper gives a detailed system description of our submission in SemEval 2019 Task 9 Subtask A. We transfer Self-Attention Network (SAN), a successful model in machine reading comprehension field, into this task. Our model concentrates on modeling long-term dependency which is indispensable to parse long and complex sentences. Besides, we also adopt techniques, such as contextualized embedding, back-translation, and auxiliary loss, to augment the system. Our model achieves a performance of F1=76.3, and rank 4th among 34 participating systems. Further ablation study shows that the techniques used in our system are beneficial to the performance.

1 Introduction

Suggestion mining is a trending research domain that focuses on the extraction of extract tips, advice, and recommendations from unstructured text. To better recognize suggestions, instead of only matching feature words, one must have the ability to understand long and complex sentences.

SemEval-2019 Task 9 provides the suggestion mining task (Negi et al., 2019). The task can be recognized as a text classification task, given a sentence collected from user feedback, participating systems are required to give a binary output by marking it as suggestion or non-suggestion.

To address this problem, we focus on solving long-term dependency on long and complex sentences. Consequently, we transfer Self-Attention Network (SAN), a successful model in machine reading comprehension field, in which long-term dependency is crucial, into this task. Furthermore, we also utilize multiple techniques to improve the suggestion mining system.

2 System description

In this paper, we consider suggestion mining as a text classification task. Figure 1 gives an overview of our model. First, the input text is converted into word embeddings with linguistic features. Then, we use several stacked semantic encoders to generate the hidden representations for each token. On top of that, a softmax output layer estimates the probability of the text being a suggestion.

2.1 Input encoding

The input encoding layer is in charge of encoding each token of the input text to singular vectors. Tokenization is completed during preprocessing. In our work, we adopt WordPiece embedding (Wu et al., 2016) and feed it into a pretrained language model (LM) to generate contextualized embeddings. Compared to context independent word vectors, such as widely used GloVe (Pennington et al., 2014), SGNS (Mikolov et al., 2013), contextualized vectors show significant advantages in disambiguation and sentence modeling. Besides, well-pretrained language model also transfers external knowledge to this task, full use of transfer learning is the key to the advance in modern neural natural language processing. On the choice of the pretrained language model, we compared two publicly available models ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018), and we finally choose BERT for better performance. Due to the out-of-memory issues [1], we do not update the parameters of BERT during training, thus we only use it as a static feature extractor.

[1] https://github.com/google-research/bert#out-of-memory-issues

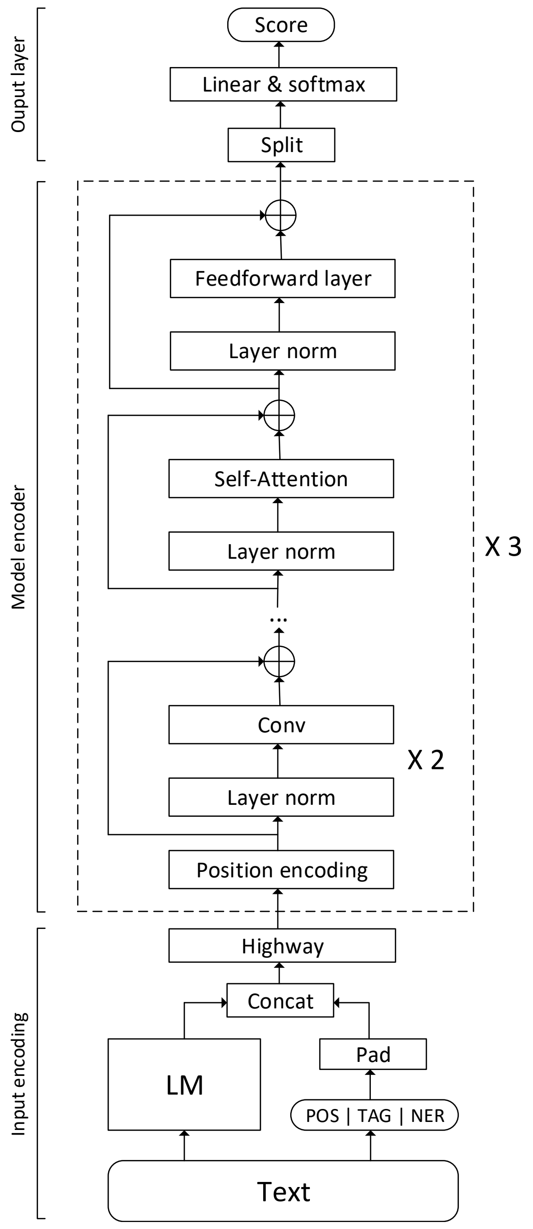


Figure 1: Overview of our system

Also, linguistic features are extracted to improve system performance further. In this work, we extract part-of-speech (POS) and named entities (NER) by spaCy [2]. Two kinds of part-of-speech granularity are used, primary POS and extended POS tag (TAG). In order to obtain linguistic feature sequences with the same length as the BERT outputs, we pad zero vectors at the start and end position of the linguistic feature sequences. Then, the contextualized vectors and linguistic feature vectors are concatenated. A two layers highway network (Srivastava et al., 2015) is also adopted on top of this representation. The vectors are projected to d dimensions immediately.

[2]https://spacy.io

2.2 Model encoder

The model encoder is the central part of our system, and it is in charge of modeling long-term dependency and extracting deep features. Because of the success of Self-Attention Network (SAN) (Shen et al., 2018) in various NLP tasks, we adopt a structure from QANet (Yu et al., 2018), which is a variant of Self-Attention Network, as our model encoder.

As is shown in the middle part of Figure 1, the model encoder is a combination of convolution, self-attention and feed-forward layer. This structure is repeated three times. The input vectors of this structure are firstly added by sinusoidal position embedding (Gehring et al., 2017) to encode a notion of the order in the sequence. The position embedding is calculated as follows:

$$\begin{aligned} PE_{i,2j} &= sin(i/10000^{2j/d}) \\ PE_{i,2j+1} &= cos(i/10000^{2j/d}) \end{aligned} \tag{1}$$

After that there are convolution blocks, following (Yu et al., 2018), depth-wise separable convolution layers are chosen for better generalization and memory efficiency. The model encoder is highly dependent on input layer normalization and residual connections, each block is in a uniform structure: *layer normalization / operation / residual connection*. In our system, we repeat convolution blocks two times for better and deeper local feature extraction.

In the self-attention layer, the scaled dot-product attention is computed:

$$A_i = softmax(\frac{QK^T}{\sqrt{d}}) \cdot V \tag{2}$$

Where Q, K, and V are query, key, and value respectively, they are the linear projection of each position in the input. As in (Vaswani et al., 2017), multi-head attention mechanism is adopted which integrates multiple scaled dot-product attentions.

$$A = [A_1; \cdots ; A_i; \cdots ; A_n] \cdot W^A \tag{3}$$

where A_i denotes the i-th head, $[\cdot]$ is concatenation operator, W^A is a trainable parameter.

At last, there is a fully connected block. In our method, it is a little different from the original work (Yu et al., 2018). We append a gate mechanism to refine tokens by their importance (Wang et al., 2017).

$$S = FFN^2(LayerNorm(H)) \odot G + H \quad (4)$$
$$G = G^*/max(G^*) \quad (5)$$
$$G^* = sigmoid(LayerNorm(H) \cdot W^G + b^G) \quad (6)$$

where we assume the input of this block is $H \in \mathbb{R}^{m\times d}$, m is the sequence length, FFN^2 denotes a 2-layer non-linear feed-forward network, $S \in \mathbb{R}^{m\times d}$ represents the output of this fully connected block. $G \in \mathbb{R}^{m\times 1}$ and $G^* \in \mathbb{R}^{m\times 1}$ are the output weight of the gate. $W^G \in \mathbb{R}^{m\times 1}$ and $b^G \in \mathbb{R}^1$ are trainable parameters. The maximum operation aims to select the maximum element in G^*, and the division operation normalizes these weights so that the maximum weight in G is always one.

2.3 Output layer

Given the output $S = [s_1, s_2, \cdots, s_m]$ of previous layers, this output layer converts these hidden representations into the final probability. Since we have adopted BERT in the input encoding layer, the first vector of the sequence is a special classification token [CLS], which can be used as the representation of the whole sentence. We split the matrix S and take the first vector $s_1 \in \mathbb{R}^d$. The probability of the input text being a suggestion text is estimated as follows.

$$p = softmax(W^p \cdot s_1 + b^p) \quad (7)$$

where $W^p \in \mathbb{R}^{2\times d}$ and $b^p \in \mathbb{R}^2$ are trainable parameters, $p \in \mathbb{R}^2$ denotes the output probabilities including "yes" probability p^1 and "no" probability p^0.

2.4 Loss

We treat this task as a text classification problem and use log-loss as the loss function.

$$\mathcal{L}_0 = \frac{1}{N}\sum_{i=1}^{N} y_i log(p_i^1) + (1-y_i)log(p_i^0) \quad (8)$$

where N is the number of examples, $y_i \in \{0,1\}$ represents the label of i-th example, p_i^1 and p_i^0 are the predictions.

Besides, in order to better recognize important tokens and filter trivial tokens out, we add an auxiliary loss to discourage large weights in G in Equation 4. Thus only those tokens that have contributions to the classification have non-zero weights in G.

$$\mathcal{L}_1 = \beta\frac{1}{N}\sum_{i=1}^{N} \|G_i\|_1 \quad (9)$$

where $\|\cdot\|_1$ denote 1-norm, β is a hyper-parameter, we use $\beta = 10^{-3}$ in this work. The final loss is the sum of $\mathcal{L}_0$ and $\mathcal{L}_1$.

2.5 Class imbalance

As is pointed out in (Negi et al., 2019), suggestions appear sparsely in online reviews and forums, and this makes class imbalance a critical problem. For simplicity, we do not take measures during training. In inference, we slightly adjust the predicted probability and divide it by a *priori*, which is the rate of positive examples in training data.

$$\begin{aligned} p &= softmax((W^p \cdot s_1 + b^p) \times 10) \\ p^{1*} &= p^1/priori \\ p^{0*} &= p^0/(1-priori) \end{aligned} \quad (10)$$

where p^{1*} and p^{0*} are the actual predictions for inference, the system outputs "yes" when p^{1*} is larger than p^{0*}. Other symbols are the same as in Equation 7.

2.6 Back-translation

Because the given training data set is not large, we also utilize a data augmentation technique to enrich the training data. The data augmentation technique we used is back-translation (Yu et al., 2018). Specifically, we first translate the given training data into Chinese by a neural machine translation system and then translate it back into English by another neural machine translation system. The two neural machine translation systems are trained on a subset of the WMT18 data sets [3]. Both original training data and the augmented data are applied to train our text classification system, but the augmented data is given a small weight (=0.2) when calculating the loss.

3 Experiments

3.1 Setup

SemEval 2019 Task 9 Subtask A [4] provides a suggestion mining data set collected from feedback

[3]http://statmt.org/wmt18
[4]https://competitions.codalab.org/competitions/19955

posts on Universal Windows Platform. The data set is split into an 8980-example training set, a 592-example validation set, and an 833-example test set. Since the organizer does not limit the usage of the validation set, we merge it into training data and train our model through the k-fold cross-validation method. Specifically, we split all training data into eight subsets, and guarantee the rate between the number of positive examples and the number of negative examples is about 1:3 in each subset. The preprocessing process is implemented as the description in section 2.1.

The kernel size of convolution layers is 7, and the hidden size d is 256, the number of heads is 8 in multi-head self-attention layers. Adam optimizer (Kingma and Ba, 2014) with learning rate 0.0008 is used for tuning the model parameters. The mini-batch size is 32. For regularization, the dropout rate is set to 0.1. The submission predictions are obtained by integrating eight runs through voting.

Approach	Test F1	
Baseline	26.8	
1st ranked system	78.1	
2nd ranked system	77.8	
3rd ranked system	77.6	
Our system	76.3	
5th ranked system	74.9	
Ablation - single model	F1	Δ
Our full system	73.3	-
- Contextualized embedding	67.6	-5.7
- Back-translation	71.4	-1.9
- Priori	72.5	-0.8
- Linguistic features	72.6	-0.7
- Auxiliary loss	72.7	-0.6

Table 1: Performance of the Top 5 systems on the leaderboard of subtask A, and our ablation experiments.

3.2 Results

Table 1 shows the main results on the test set. Compared with the rule baseline, participants improve their performance with substantial gains. Our system achieved F1=76.3 on the test set and ranked 4th among all 34 teams.

In order to evaluate the individual contribution of each feature, we run an ablation study. Contextualized embedding is most critical to the performance, and it concludes that transferring common sense by learning large corpus is vital for this task. Back-translation accounts for about 2% of the performance degradation, which clearly shows the effectiveness of data augmentation. Besides, linguistic features, auxiliary loss, and *priori* are also beneficial to the system.

Base		Large	
F1	F1	F1	F1
72.9	71.8	75.4	75.9
72.9	70.6	73.9	76.6
74.3	74.8	74.0	76.2
75.3	73.5	76.3	75.1
Ensemble			
#	F1	#	F1
2	72.9	8	74.5
4	74.7	12	75.5
16	76.3	Oracle	79.6

Table 2: Performance of every single model and ensemble models.

3.3 Ensemble models

Table 2 reports the effect of ensemble. To obtain the submission predictions, we trained eight models on the eight subsets mentioned in section 3.1. Four of these models are based on a 110M parameters base BERT model, and the other four are based on a 340M parameters large BERT model. The performance of every single model is reported. It is evident that different data partition causes quite a large performance variance. Thus, ensemble is necessary. The result shows that as the number of models increases the ensemble effect improves. Also, we experiment by searching the optimal model combination assuming the test label is known to show the performance limitation (the Oracle performance).

4 Conclusion

In this work, we adopt multiple techniques to improve a suggestion mining system. The core of our system is a variant of Self-Attention Network (SAN), which originates from the machine reading comprehension field. Based on this model, techniques, such as contextualized embedding, back-translation, linguistic features, and auxiliary loss, are investigated to improve the system performance further. Experimental results illustrate the effect of our system. Our model achieves a performance of F1=76.3, and rank 4th among 34 participating systems.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. 2018. Bi-directional block self-attention for fast and memory-efficient sequence modeling. *arXiv preprint arXiv:1804.00857*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

YNU_DYX at SemEval-2019 Task 9: A Stacked BiLSTM Model for Suggestion Mining Classification

Yunxia Ding, Xiaobing Zhou*, Xuejie Zhang
School of Information Science and Engineering
Yunnan University, Yunnan, P.R. China
*Corresponding author: zhouxb@ynu.edu.cn, ynudyx@gmail.com

Abstract

In this paper we describe a deep-learning system that competed as SemEval 2019 Task 9-SubTask A: Suggestion Mining from Online Reviews and Forums. We use Word2Vec to learn the distributed representations from sentences. This system is composed of a Stacked Bidirectional Long-Short Memory Network (SBiLSTM) for enriching word representations before and after the sequence relationship with context. We perform an ensemble to improve the effectiveness of our model. Our official submission results achieve an F_1-score 0.5659.

1 Introduction

Suggestions in the Oxford Dictionary are defined as ideas or plans for consideration. Some of the listed synonyms of suggestions are proposal, proposition, recommendation, advice, hint, tip, clue. In general, other types of text and suggestions are easily distinguished by the definition of the suggestion (Negi and Buitelaar, 2015).

Suggestion mining can be defined as the extraction of suggestions from unstructured text, where the term 'suggestions' refers to the expressions of tips, advice, recommendations etc. We often see comments on products in product forums which are recommended or not recommended, and some users will consider whether to purchase the product based on these comments. Suggestion mining is also defined as automatic extraction of recommendations from a given text. These texts that express user suggestions can usually be found in social media platforms, blogs, or product online forums (Negi and Buitelaar, 2017; Negi et al., 2016).

Suggestion mining remains a relatively young area compared to Sentiment Analysis (Negi and Buitelaar, 2017), due to the lack of a large number of tagged datasets. SemEval 2019 Task 9-SubTask A is mainly a binary classification, identifying sentences which express suggestions in a given text. And we need to classify each sentence of given text, the categories being suggestions or non suggestions. This is similar to the polarity analysis of emotions, as positive or negative instances, respectively.

In the past, classification problems in natural language processing were solved by traditional methods, such as sentiment analysis (Nielsen, 2011; Go et al., 2009; Bollen et al., 2011; Mohammad et al., 2013; Kiritchenko et al., 2014) which were handled by classifiers such as Naive Bayes (McCallum et al., 1998)and SVMs (Gunn et al., 1998). However, deep neural networks achieve increasing performance compared to traditional methods, due to their ability to learn more abstract features from large amounts of data, producing state-of-the-art results in sentiment analysis.

The SubTask-A is part of SemEval 2019 Task9: Suggestion Mining from Online Reviews and Forums, and is concerned with classifying suggestions forum for Windows platform developers—suggestions or non suggestions. There are 34 teams who participated in the task(Negi et al., 2019).

In this paper we describe our system designed for this task. First, we model the sentence and establish the vector representation of the sentence through Word2Vec (Mikolov et al., 2013a), a Stacked Bidirectional Long-Short Memory Network(SBiLSTM) for enriching word representations with context. Finally, the sentence representation is projected into the label space through a Dense Layer.

The rest of the paper is organized as follows: Section 2 provides the details of the proposed model; Data Processing and analysis are discussed in section 3. Experiments and results are described in Section 4. Finally, we draw conclusions in Section 5.

2 System Description

2.1 Network Architecture

First we use the embedding layer to get a distributed representation of the words, then feed the results of the embedding layer to the first BiLSTM layer. Using the LSTM model (Hochreiter and Schmidhuber, 1997) can better capture long-distance dependencies and learn what information to remember and what information to forget by training the model. BiLSTM captures the semantic information of sentences from both forward and reverse directions. In order to get more fine-grained sentence information, we use 2-layer BiLSTM. The features obtained from the first BiLSTM are then put into the next BiLSTM (Graves and Schmidhuber, 2005; Graves et al., 2013). The final result is obtained by the softmax used as the activation function in the Dense layer. The model architecture is show in Figure 1.

2.2 Word Embedding

Word embedding is unarguably the most widely known technology in the recent history of NLP. It converts words into a distributed representation that can solve dimensional disaster problems (Bengio et al., 2003). And it projects words from high-dimensional space to a lower-dimensional vector space through hidden layers and performs semantic feature extraction (Kim, 2014). This technology has a wide range of applications in NLP. It is well-known that using pre-trained embedding helps, as well.

Word embeddings can better measure the similarity between words, and are also dense vector representation of words that capture semantic and syntactic information. So in this task we try to use the Word2Vec (Mikolov et al., 2013b) and Glove (Pennington et al., 2014) to get the vector representations of the words.

3 Data Processing and Analysis

There are two categories: suggestion and non-suggestion in the data set given in the shared task. And the organizer provides the third version data sets: a total of 8,500 sentences in training and 592 sentences in trial and 833 sentences in test.

3.1 Data Processing

We perform a series of specification processing on the text in the dataset.

- All characters are converted to lowercase.
- Contraction normalization, like replacing "don't" and "dont" with "do not", "cant" and "can't" with "can not" and so on.
- All hyperlinks are replaced by "url".

After the above processing, we find that some words in the text have not been segmented correctly. For example, the correct form of "supportedcultures" should be "supported cultures". There are many such words in the dataset, and if we don't deal with them, there will be a lot of unknown words in the vocabulary. In order to solve this problem, we use *Ekphrasis* (Baziotis et al., 2017), a tool geared towards text from social networks, such as Twitter or Facebook. *Ekphrasis* performs tokenization, word normalization, word segmentation (for splitting hashtags) and spell correction.

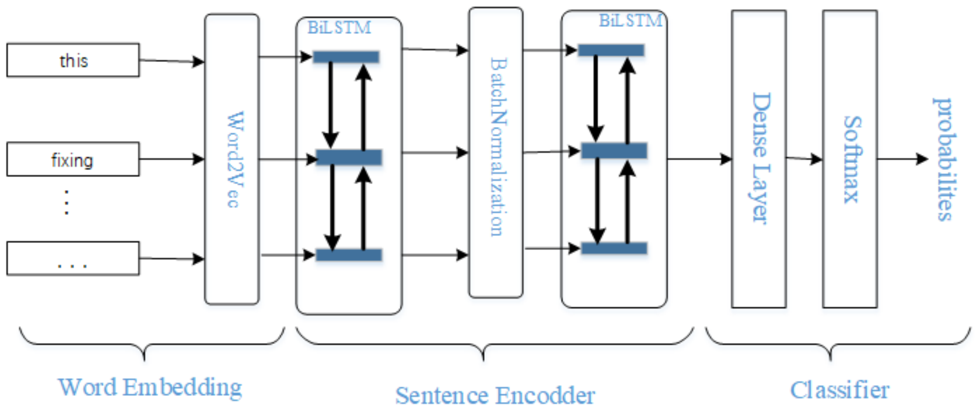


Figure 1: Our system architecture

3.2 Data Analysis

Sentence length: In order to determine the length of the training set sentence in the input model, after the data processing is finished, we analyze the length of each sentence. First, we find that the longest sentence is 495, the shortest is 0, and the sentence length is shown in Figure 2.

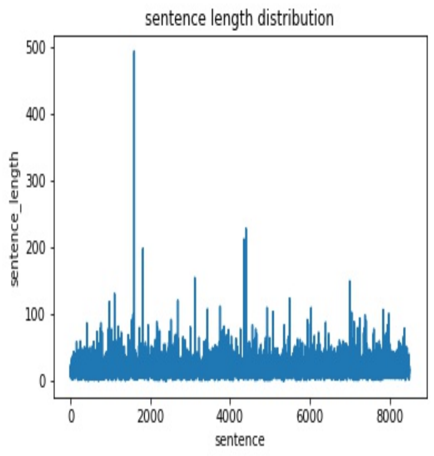


Figure 2: Training set sentence length distribution

If the sentence is too long, the calculation time will increase. If it is too short, the extra information will be lost. Therefore, according to the sentence length distribution map, the length of the sentence in the input model is finally determined to be 75.

Training set label: Table 1 shows the label distribution for the dataset.

	Train set	Trial set	Test set
Suggestions	2085	296	87
Non-suggestions	6415	296	746

Table 1: Number of sentences in each dataset.

It can be seen from Table 1 that the label of the training set is extremely unbalanced, and the ratio of suggestion and non-suggestion reaches 1 : 3. In order to balance the training set data, we process those sentences labeled with the suggestions. We use the shuffle data enhancement method, which re-range the word order inside the sentences. We performed two shuffle operations, and the last data in the suggestions and non-suggestions in the final dataset were 6255 and 6415, respectively.

4 Experiments and Results

We use Python based neural network library, Keras[1], for the implementation. We train and validate our models on the training and validation sets provided by the organizer. The official evaluation metric is based on macro average F_1-*score* measure. More details about the data and the evaluation metrics can be found in the task description paper (Negi et al., 2019).

SBiLSTM: For the Stacked BiLSTM, the first layer BiLSTM *units* = 256, and the second layer BiLSTM *units* = 180.

Optimization: Optimization is carried out with Adaptive Moment Estimation(Adam) (Kingma and Ba, 2014), using the default learning rate 0.001, and hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$.

Loss Function: Usually the multi-classification problem uses *categorical crossentropy* as the loss function. But our system uses *binary crossentropy* in this binary classification.

As shown in Table 2, the number of unknown words in the dataset in Word2Vec are less than those in Glove. To reduce the number of unknown words in the embedding, making the context semantics better learned by the model. We randomly assign the vectors of unknown words. And we experimented with the embedding of the words Word2Vec and Glove, and found that the results of Word2Vec performed better than Glove.

Embedding	Ukw	Evaluation set F1
Glove	394	0.5422
Word2Vec	231	**0.5482**

Table 2: Comparison between Word2Vec and Glove on DBiLSTM models.

We compare the two network structures of Stacked LSTM and Stacked BiLSTM. As can be seen from the results in Table 3, the performance of the Stacked BiLSTM is better than that of LSTM.

Model	Embedding	Evaluation set F1
LSTM	Word2Vec	0.5612
BiLSTM	Word2Vec	**0.5637**

Table 3: Comparison of LSTM and BiLSTM.

Finally, we train the single model with the dropouts of 0.55, 0.60, 0.65, respectively. Each

[1]http://keras.io/

single model produces a soft probability, then we use the sum of the probabilities as the final prediction. We find that the performance of ensemble model is better than a single model.

Dropout	Evaluation set F1
0.55	0.5307
0.60	0.5214
0.65	0.5422
Ensemble	**0.5659**

Table 4: The model adopts Word2Vec, data enhancements and Stacked BiLSTM architecture. Dropout is recurrent-dropout in the BiLSTM layer.

5 Conclusion and Future Work

In this paper, we have presented a Stacked BiLSTM(SBLSTM) model for predicting the suggestion mining classification. The word embedding Word2Vec is used in our system, an ensemble method can significantly enhance the overall performance.

In the future, we will try to use language models to obtain the representation of sentences, and explore other NLP models to make the experimental results better. At the same time, we will also try to use transfer learning technology.

Acknowledgments

This work was supported by the Natural Science Foundations of China under Grant Nos. 61463050, 61702443 and 61762091, and the Project of Innovative Research Team of Yunnan Province under Grant No. 2018HC019.

References

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada. Association for Computational Linguistics.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Johan Bollen, Huina Mao, and Alberto Pepe. 2011. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. *Icwsm*, 11:450–453.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12).

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Steve R Gunn et al. 1998. Support vector machines for classification and regression. *ISIS technical report*, 14(1):5–16.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. 2014. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, 50:723–762.

Andrew McCallum, Kamal Nigam, et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *arXiv preprint arXiv:1308.6242*.

Sapna Negi, Kartik Asooja, Shubham Mehrotra, and Paul Buitelaar. 2016. A study of suggestions in opinionated texts and their automatic detection. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 170–178.

Sapna Negi and Paul Buitelaar. 2015. Towards the extraction of customer-to-customer suggestions from reviews. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2159–2167.

Sapna Negi and Paul Buitelaar. 2017. Inducing distant supervision in suggestion mining through part-of-speech embeddings. *arXiv preprint arXiv:1709.07403*.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Finn Årup Nielsen. 2011. A new anew: Evaluation of a word list for sentiment analysis in microblogs. *arXiv preprint arXiv:1103.2903*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

YNU-HPCC at SemEval-2019 Task 9: Using a BERT and CNN-BiLSTM-GRU Model for Suggestion Mining

Ping Yue, Jin Wang and Xuejie Zhang
School of Information Science and Engineering
Yunnan University
Kunming, P.R. China
Contact: xjzhang@ynu.edu.cn

Abstract

Consumer opinions towards commercial entities are generally expressed through online reviews, blogs, and discussion forums. These opinions largely express positive and negative sentiments towards a given entity; however, they may also contain suggestions for improving the entity. In this task, we extract suggestions from a given unstructured text, in contrast to the traditional opinion mining systems. This type suggestion mining is more applicable and extends capabilities. In this paper, we propose the use of bidirectional encoder representation learned from transformers (BERT) to address the problem of domain specific suggestion mining in task A. In detail, BERT is also used to extract feature vectors and perform fine-tuning for the task . For Task B, we applied an ensemble model to combine the BiLSTM, CNN, and GRU models, which can perform cross domain suggestion mining. Officially released results show that our system performs better than the baseline algorithm does.

1 Introduction

Suggestion mining is used to extract advice from text such as that provided in online reviews, blogs, discussion forums, and social media platforms where consumers share their opinions towards commercial entities like brands, services, and products. Most of the traditional sentiment analysis methods are emotion classifications. Opinion mining can improve service and quality. Such systems have become an effective way for marketing, economics, politics, and advertising. The application of suggestion mining provides the motivation, for the SemEval 2019 Task 9 (Negi et al., 2019), which contains two subtasks that classify given sentences into suggestion and non-suggestion classes. Subtask A requires a system to achieve domain specific training, whereby the test dataset will belong to the same domain as the training and development datasets. This was part of a suggestion forum for windows platform developers. Subtask B applies the system to cross domain training, where training, development, and test datasets will belong to different domains. Training and development datasets will remain the same as Subtask A, while the test dataset will belong to the domain of hotel reviews.

There are many methods in sentiment analysis. In many reports on this subject, it has been implied that these models help improve classification. Successful models include convolutional neural networks (CNN), long short-term memory (LSTM), and bi-directional LSTM (BiLSTM). CNN can capture local n-gram features, while LSTM can maintain memory in the pipelines and solve the problem of long sequence dependence in neural networks.

In this paper, we propose a bidirectional encoder representation learned from transformers (BERT) model (Devlin et al., 2018) for Task A. It comprises two phases. The first phase is called pre-training and is similar to word embedding. The second phase is called fine-tuning and uses a pre-trained language model to complete specific NLP downstream tasks. We used a pre-trained model that was provided by Google AI team. It included weights for the pre-trained model and a vocab file that maps component words of sentences to indexes of words. It also included the JSON file, which specifies the model hyper-parameters. Fine-tuning was applied to sequence classification: the BERT directly takes the final hidden state of the first [CLS] token, adds a layer of weight, and then softmax predicts the label probability. The structure is shown in Figure 2.

For Task B, we apply the bert model to test data, the score is 0.343, it is very low. the reason is that the task B is cross-domain training, so we intro-

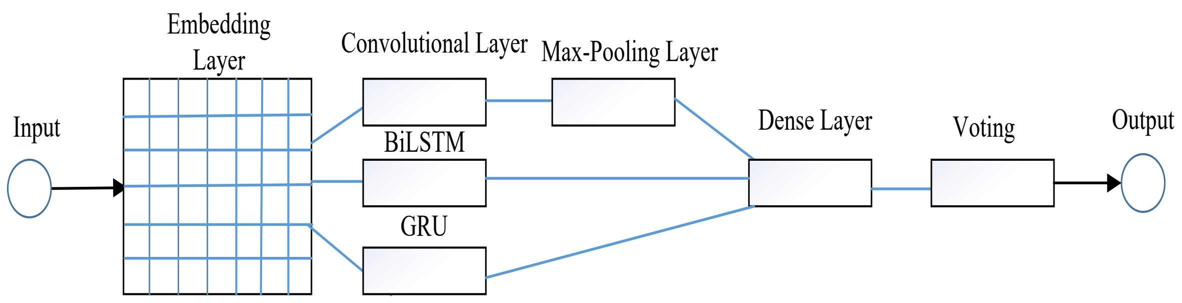


Figure 1: The CNN-BiLSTM-GRU architecture

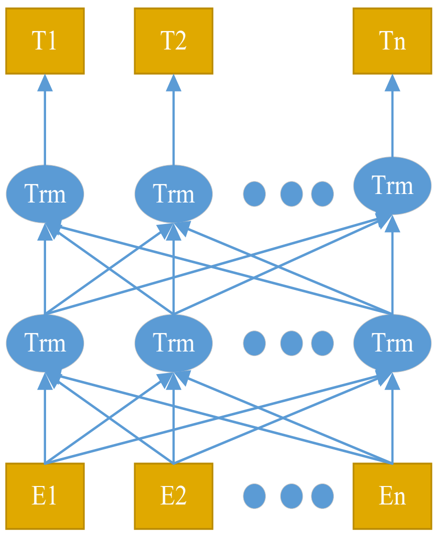


Figure 2: The architecture of BERT

duced an ensemble model that includes the CNN, BiLSTM, and GRU model. The structure is shown in Figure 1. We constructed the word vectors from 300-dimensional Glove vector. Then, a word vector matrix was loaded into the embedding layer. After this, the CNN applies the convolutional layer and max pooling layer to extract n-gram features, and passes through the dense layer to classify the sentence. BiLSTM can obtain the semantic information from the context. The forward and backward layers are connected to the output layer. GRU has a structure similar to that of LSTM, but is simpler. Finally, we combined CNN with BiLSTM and GRU using a soft-voting method, and output the results. The experimental results show that our model has good performance. According to the official review, we achieved sixth place among the 34 teams working on Task A.

The rest of the paper is organized as follows. In Section 2, we describe the BERT model. There, we also detail CNN, BiLSTM, and GRU and their combination. The comparative experimental results are presented in Section 3. Conclusions are drawn in Section 4.

2 The BERT and CNN-BiLSTM-GRU model

Figure 2 shows the BERT model. First, for each token, a representation is generated by its corresponding token embedding, segment embedding, and position embedding. Word-Piece was embeded (Wu et al., 2016) along with 30,000 token vocabularies. Finally, an output layer was used to fine-tune the parameters. Figure 1 shows the ensemble model used to combine the CNN, BiLSTM, and GRU models. First, all component words were transformed to a feature matrix by an embedding layer. Then a convolutional layer and a max pooling layer, were used for feature extraction. To avoid over-fitting, a dropout layer was used after both convolution and max-pooling layers. BiLSTM outputs predictive label sequence directly to input sentences. GRU is a variant of LSTM that has fewer parameters and is relatively easier to train. We embedded these models with the vote method and finally output the result.

2.1 Bidirectional Encoder Representations from Transformers (BERT)

Input characterization. For the task of sentence classification, BERT will add the [CLS] and [SEP] identifiers to the beginning and end of each input text; thus, the maximum sequence length can be described as follows.

$$max\_seq = S_t + 2 \qquad (1)$$

where *max_seq* denotes the maximum sequence length, and *St* is the set text length. We set the S_t=78, and *max_seq*=80. For every input sentence, BERT introduces masked language mode, and next sentence prediction. Input embedding is the sum of token embedding, segmentation embedding, and position embedding.

Transformer. The multi-layer transformer Vaswani et al. (2017) structure operates through the attention mechanism to convert the distance between two words at any position into the numeral 1. Owing to the transformer's overall architecture, the input sequence will first be converted into a word embedding vector, which can be used as the input of the multi-head self-attention module after adding the position coding vector. The output of the module can be used as the output of the encoder module after passing through a fully connected layer.

Output.The highest hidden layer of [CLS] is directly connected to the output layer of softmax as a sentence. The output result of BERT is label probability. The sum of the probabilities of all labels is 1, and the probability value of returning a label is the same as the order of setting labels in MrpcProcessor. This task sets the labels to 0 and 1. The probability of the first column returned in this experiment is 0, and the probability of the second column is 1.

2.2 CNN

Embedding Layer. The embedding layer is the first layer of model. Load Glove (Lee et al., 2016; Cun et al., 1990) with word embedding (Zahran et al., 2015) and is used for model initialization of online reviews. The embedded layer converts a positive integer (subscript) into a vector of fixed size N. N is defined as 80; any sentence exceeding this size is reduced to 80, and any sentence with a size less than 80 is padded to 80 by adding 0s.

Convolution Layer. The convolution layer is used to extract n-gram features from the embedding matrix. The calculation method of the convolution layer is as follows,

$$conv = \sigma(Mat \circ W + b) \qquad (2)$$

where σ is an activation function, *Mat* indicates an embedding matrix, W and b respectively denote convolution kernel and bias. Here, $\circ$ is a convolution operation. We use 3*3 convolution kernels. The activation function is ReLU (Nair and Hinton, 2010)

Max pooling layer. Pooling is selecting a part of the input matrix and is used to choose the best representative for the region. The max pooling layer selects the max feature.

Dropout Layer. To avoid over-fitting, we introduce the dropout layer (Hinton et al., 2012) after both a convolution layer and max pooling layer, which is to randomly throw away some weight of the current layer. It can reduce model complexity and enhance the generalization ability of the model.

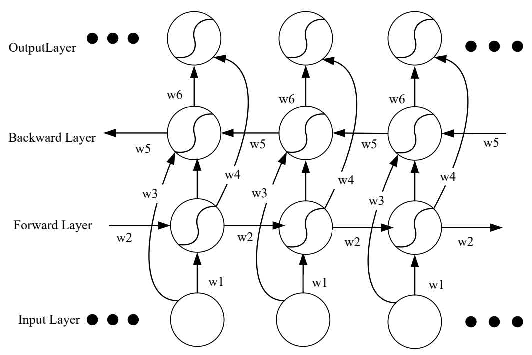


Figure 3: The architecture of BiLSTM

2.3 BILSTM and GRU

Bi-directional Long Short-Term Memory (BiLSTM) (Brueckner and Schuller, 2014; Li and Chang, 2015) is a variant of Recurrent Neural Network (RNN). Owing to its design characteristics, BiLSTM is ideal for modeling time-series data such as text data. Figure 3 shows the BiLSTM structure. BiLSTM is an abbreviation of LSTM Graves (2012); Greff et al. (2016); Graves (2012), which is a combination of forward LSTM and backward LSTM. Both are often used to model context information in natural language processing tasks.

Gated Recurrent Unit (GRU)(Cho et al., 2014) is a variant of LSTM, although the model is simpler than the standard LSTM model. It combines a forget gate and input gate into a single update gate. It also mixes cell state with hidden state.

$$\begin{aligned} z_t &= \sigma(W_z \cdot [h_{t-1}, x_t]) \\ r_t &= \sigma(W_r \cdot [h_{t-1}, x_t]) \\ \tilde{h}_t &= \tanh(W \cdot [r_t * h_{t-1}, x_t]) \\ h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t) \end{aligned} \qquad (3)$$

where h_t is hidden states, x_t is the input vector, σ is the sigmoid function, and r_t and z_t are the reset and update doors, respectively.

2.4 Ensemble

Each classifier is independently classified, and the integrated model can improve the correct rate. In this task, each base learner has a predicted value, and we used a soft-voting classifier as the final predicted value. The soft-voting classifier predicts the class label based on the sums of the predicted probabilities. The corresponding type with the highest probability is the final prediction result.

Model	Trial	Test
CNN	0.505	0.216
BiLSTM	0.498	0.180
CNN-BiLSTM	0.667	0.210
BERT	**0.851**	**0.735**

Table 1: The experiment results.

Parameters	BiLSTM/GRU	CNN
Neurons	60	120
Dropout rate	0.4	0.0
Weight	2	5
Activation	softmax	softmax
Init mode	LeCun	LeCun
Learning rate	0.001	0.2
Momentum	0.4	0.4

Table 2: The best-tuned parameters.

3 Experiments and Evaluation

In this section, we report the experiments were conducted to evaluate the proposed models on both sub-tasks. We also report the results of the official review. The details of the experiment are described as follows.

3.1 Data Preparation

Subtask A. Organizers provided training data from online forum comments that included three parts: id, sentence, and label. The given label is 0, indicating that the suggestion is not recommended. Here, 1 indicates a positive suggestion. This is equivalent to the suggestion mining used to discover sentences with suggestions. The positive and negative emotional statements of a given data set are unbalanced, and the negative emotions are three times more abundant than the positive emotions. According to this situation, we used oversampling to process the data; the positive emotional sentences were randomly copied from the training set in the same proportion as the negative emotional sentences. We extracted 0.2 ratio data as a validation set in the training set. In this experiment, we used the BERT-Base model which is pre-trained by Google AI team to process the text.

Subtask B. To address the problem of imbalance in data distribution, Task B uses the define loss function. In our model, we introduced a focal-loss function (Lin et al., 2017) that reduced the weight of many negative samples in training. This loss function is a dynamically scaled cross entropy loss function. As the correct classification increases, the scale factor in the function is reduced to zero. This scale factor can automatically reduce the impact of simple samples during training. Quickly focus your model on difficult samples. Data processing removes stop words, replaces URLs with <urls>, and removes characters except for alphanumeric characters and punctuation.

3.2 Implementation Details

Subtask A. In this experiment, TensorFlow (GPU backed) was used. We used the BERT-Base model to process the data. We introduced other three models (CNN, BiLSTM, and BiLSTM) as baseline algorithms. We combined commonly used parameters to tune-in the training. For the task, the batch size was 30, the learning-rate set was 2e-5, and the number of the training epoch was 10.

Subtask B. We used Scikit-Learn to perform a grid search (Pedregosa et al., 2013) to tune the hyper-parameters, by which we could find the best parameters to evaluate the system. The weight indicates the weight_constraint. The LeCun indicates LeCun uniform. The fine-tuned parameters are summarized in Table 2.

3.3 Evaluation Metrics

Classification performance of the submissions will be evaluated based on binary F_1-score for the positive class. Binary F_1-score will range from 1 to 0.

3.4 Results and Discussion

The trial and test data for the baseline model and the BERT model shows that our model has the best score in Table 1.

Subtask A. Our system achieved the F1 score of 0.7353 on Subtask A, and the baseline score was 0.2676. The results show that our proposed system is a significant improvement over the baseline. The main reason is that not only the BERT is a multi-layer bidirectional transformer encoder, but also the BERT-Base model is powerful pre-training model.

Subtask B. Our model score was 0.5035, while the baseline score was 0.7329. There is a need to do more for improvement. For cross-domain suggestion mining, it is necessary to increase the generalization ability of the training model to achieve use in multiple domains.

4 Conclusion

In this paper, we describe a task system that we submitted to SemEval-2019 for suggestion mining. For Subtask A, we use the BERT model. For Subtask B, we introduced CNN combined with BiLSTM and GRU. The experimental results show that the models we introduced achieved good performance in the final evaluation phase. In future research, we will attempt to generalize models with better capabilities to obtain more better results.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants No.61702443 and No.61762091, and in part by Educational Commission of Yunnan Province of China under Grant No.2017ZZX030. The authors would like to thank the anonymous reviewers and the area chairs for their constructive comments.

References

Raymond Brueckner and Bjrn Schuller. 2014. Social signal classification using deep blstm recurrent neural networks. In *IEEE International Conference on Acoustics*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. 1990. Handwritten digit recognition with a back-propagation network. *Advances in Neural Information Processing Systems*, 2(2):396–404.

Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Alex Graves. 2012. *Long Short-Term Memory*.

K Greff, R. K. Srivastava, J Koutnik, B. R. Steunebrink, and J Schmidhuber. 2016. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks & Learning Systems*, 28(10):2222–2232.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Yang Yin Lee, Hao Ke, Hen Hsen Huang, and Hsin Hsi Chen. 2016. Less is more: Filtering abnormal dimensions in glove. In *International Conference Companion on World Wide Web*.

Tianshi Li and Baobao Chang. 2015. Semantic role labeling using recursive neural network.

Tsung Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. 2017. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, PP(99):2999–3007.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *International Conference on International Conference on Machine Learning*.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Fabian Pedregosa, Gal Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, and Vincent Dubourg. 2013. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(10):2825–2830.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Cao Yuan, Gao Qin, and Klaus Macherey. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.

Mohamed A. Zahran, Ahmed Magooda, Ashraf Y. Mahgoub, Hazem Raafat, Mohsen Rashwan, and Amir Atyia. 2015. *Word Representations in Vector Space and their Applications for Arabic*.

Zoho at SemEval-2019 Task 9: Semi-supervised Domain Adaptation using Tri-training for Suggestion Mining

Sai Prasanna
Zoho
saiprasanna.r@zohocorp.com
sai.r.prasanna@gmail.com

Sri Ananda Seelan
Zoho
anandaseelan.ln@zohocorp.com

Abstract

This paper describes our submission for the SemEval-2019 Suggestion Mining task. A simple Convolutional Neural Network (CNN) classifier with contextual word representations from a pre-trained language model is used for sentence classification. The model is trained using tri-training, a semi-supervised bootstrapping mechanism for labelling unseen data. Tri-training proved to be an effective technique to accommodate domain shift for cross-domain suggestion mining (Subtask B) where there is no hand labelled training data. For in-domain evaluation (Subtask A), we use the same technique to augment the training set. Our system ranks thirteenth in Subtask A with an F_1-score of 68.07 and third in Subtask B with an F_1-score of 81.94.

1 Introduction

Task 9 of SemEval-2019 (Negi et al., 2019) focuses on mining sentences that contain suggestions in online discussions and reviews. Suggestion Mining is modelled as a sentence classification task with two Subtasks:

- Subtask A evaluates the classifier performance on a technical domain specific setting.
- Subtask B evaluates the domain adaptability of a model by doing cross-domain suggestion classification on hotel reviews.

We approached this task as an opportunity to test the effectiveness of transfer learning and semi-supervised learning techniques. In Subtask A, the high class imbalance and relatively smaller size of the training data made it an ideal setup for evaluating the efficacy of recent transfer learning techniques. Using pre-trained language models for contextual word representations has been shown to improve many Natural Language Processing (NLP) tasks (Peters et al., 2018; Ruder and Howard, 2018; Radford, 2018; Devlin et al., 2018). This transfer learning technique is also an effective method when less labelled data is available as shown in (Ruder and Howard, 2018). In this work, we use the BERT model (Devlin et al., 2018) for obtaining contextual representations. This results in enhanced scores even for simple baseline classifiers.

Subtask B requires the system to not use manually labelled data and hence it lends itself to a classic semi-supervised learning scenario. Many methods have been proposed for domain adaptation for NLP (Blitzer et al., 2007; Chen et al., 2011; Chen and Cardie, 2018; Zhou and Li, 2005; Blum and Mitchell, 1998). We use a label bootstrapping technique called tri-training (Zhou and Li, 2005) with which unlabelled samples are labelled iteratively with increasing confidence at each training iteration(explained in Section 2.4). Ruder and Plank (2018) shows the effectiveness of tri-training for baseline deep neural models in text classification under domain shift. They also propose a multi-task approach for tri-training, however we only adapt the classic tri-training procedure presented for suggestion mining task.

Detailed explanation of the submitted system and experiments are elicited in the following sections. Section 2 describes the components of the system. Following this, Section 3 details the experiments, results and ablation studies that were performed.

2 System Description

The models and the training procedures are built using AllenNLP library (Gardner et al., 2018). All the code to replicate our experiments are public and can be accessed from `https://github.com/sai-prasanna/suggestion-mining-semeval19`.

2.1 Data cleaning and pre-processing

Basic data pre-processing is done to normalize whitespace, remove noisy symbols and accents. Very short sentences with less than four words are disregarded from training.

2.2 Word Representations

We use GloVe word representations (Pennington et al., 2014) and compare the performance improvement that we obtain with pre-trained BERT representations (Devlin et al., 2018).

2.3 Suggestion Classification

Our baseline classifier is Deep Averaging Network (DAN) (Iyyer et al., 2015). DAN is a neural bag-of-words model that is considered as a strong baseline for text classification. In DAN, a sentence representation is obtained by averaging the word level representations and is fed to a series of rectified linear unit (ReLU) layers with a final softmax layer.

A simple Convolutional Neural Network (CNN) text classifier (Kim, 2014) is used for the final submission.

2.4 Training

We use the classic tri-training procedure for label bootstrapping as mentioned in (Ruder and Plank, 2018). Consider a labelled dataset L from the source domain S and an unlabelled dataset U from the target domain T. The objective of tri-training is to label U iteratively and augment it with L. Three CNN +BERT classifiers M_1, M_2, M_3 are trained separately using subsets of L namely l_1, l_2, l_3 respectively. These subsets are obtained from L using bootstrap sampling with replacement.

The above mentioned models are used to predict labels for the unlabelled set U. Predictions which are agreed by two models is considered as a new training example for the third model in the next iteration. For example, an unlabelled sentence $U_1 \in U$ is added as a labelled example to l_1, if and only if the label for U_1 is agreed upon by both M_2 and M_3. Same way, l_2 is updated with newly labelled data if those labels have been agreed by M_1 and M_3 and so on. This constitutes a single iteration of tri-training. The procedure that is used for the training of our models is mentioned in Algorithm 1.

In this way, the original training data gets added with three newly labelled subsets which are again used for the next training iteration. At the end of each iteration, validation F_1-score is calculated by using the predictions that are obtained through a majority vote. The procedure is continued until there is no improvement in the validation score.

Algorithm 1 Tri-training

```
 1: $L \leftarrow Labelled\ Data$ , $|L| = m$
 2: $U \leftarrow Unlabelled\ Data$ , $|U| = n$
 3: for $i \leftarrow 1, 2, 3$ do
 4:     $l_i \leftarrow BootstrapSamples(L)$
 5: end for
 6: repeat
 7:     for $i \leftarrow 1, 2, 3$ do
 8:         $M_i \leftarrow Train(l_i)$
 9:     end for
10:     for $i \leftarrow 1, 2, 3,$ do
11:         $l_i \leftarrow L$
12:         for $j \leftarrow 1, n$ do
13:             if $M_p(U_j) == M_q(U_j)$
14:                 $where\ p,\ q \neq i$ then
15:                 $l_i \leftarrow l_i + \{(U_j, M_p(U_j))\}$
16:             end if
17:         end for
18:     end for
19: until no improvement in validation metrics
```

3 Experiments and Results

This section details the various experiments that were performed using the above components for our submissions.

3.1 Data

The test set provided during the trial phase of the evaluation is used as the validation data for all our experiments. For those experiments that do not involve tri-training, we only use the provided labelled data from the technical domain for training.

In Subtask B, for those experiments that involve tri-training, L is the same as mentioned above. U here is obtained in two ways:

- Unlabelled data from final test set of Subtask B.
- Unlabelled data from Yelp hotel reviews (Blomo et al., 2013).

The results reported are mean and confidence intervals of Precision, Recall and F_1-score over five runs of the same experiments with different random seeds.

Subtask A - Technical Domain

Experiment	Validation			Test		
	Precision	Recall	F$_1$	Precision	Recall	F$_1$
Organizer Baseline	58.72	93.24	72.06	15.69	91.95	26.80
DAN +glove	68.51±2.43	87.30±5.00	76.69±1.06	25.40±3.56	84.60±9.87	38.84±3.10
DAN +bert	76.06±1.31	90.27±1.71	82.55±0.50	45.80±4.49	90.80±1.75	60.82±3.99
DAN +bert w/o upsampling	79.04±2.67	83.38±2.73	81.11±0.68	55.06±6.36	83.68±2.75	66.28±4.28
CNN +bert	80.34±4.21	89.93±4.23	84.76±0.52	50.34±6.70	91.72±2.55	64.81±4.86
CNN +bert w/o upsampling	83.22±3.01	84.73±3.86	83.90±0.70	58.98±5.41	88.05±1.63	**70.58**±4.24
CNN +bert +tritrain$_{Test}$*	83.06±1.96	89.19±1.88	**86.00**±0.35	52.89±2.69	90.80±2.02	66.81±1.90

Subtask B - Hotel Reviews Domain

Experiment	Validation			Test		
	Precision	Recall	F$_1$	Precision	Recall	F$_1$
Organizer Baseline	72.84	81.68	77.01	68.86	78.16	73.21
DAN +glove	82.00±4.25	52.97±9.25	64.01±5.75	73.32±3.50	46.09±7.21	56.35±4.71
DAN +bert	89.75±2.79	65.74±8.71	75.65±5.10	78.90±4.03	64.20±8.77	70.49±4.09
DAN +bert w/o upsampling	94.26±1.87	31.73±5.73	47.31±6.27	87.98±3.41	31.09±7.17	45.62±7.47
CNN +bert	93.77±1.34	51.88±6.88	66.65±5.68	90.17±2.45	50.34±8.71	64.31±6.72
CNN +bert w/o upsampling	93.94±1.36	45.99±7.59	61.53±6.73	89.75±4.41	44.08±9.38	58.66±7.79
CNN +bert +tritrain$_{Test}$*	91.91±2.06	88.32±2.05	**90.05**±0.76	81.26±1.63	83.16±1.40	**82.19**±1.03
CNN +bert +tritrain$_{Yelp}$	88.09±0.62	87.13±0.38	87.61±0.42	78.01±5.42	86.67±3.96	81.98±2.05

Table 1: Performance metrics of different models on validation and test sets of both subtasks. Confidence intervals for the metrics are reported for five runs using different random seeds on t-distribution with 95% confidence. Upsampling is used in the training dataset unless otherwise specified. Single model from experiments with * was used for the final submission.

3.2 Input

For input representations, we use 300d GloVe vectors with dropout (Srivastava et al., 2014) of 0.2 for regularization. We also experiment with the pre-trained BERT base uncased model. The BERT model is not fine-tuned during our training. A dropout of 0.5 is applied for the 768d representations obtained from BERT.

3.3 Baseline Deep Averaging Net

Our neural baseline is Deep Averaging Net (DAN) (Section 2.3). When used with GloVe, the hidden sizes of DAN are 300, 150, 75, and 2 respectively. When BERT representations are used, the hidden sizes of the network are 768, 324, 162, and 2 respectively. We report an F_1-score of 60.82 when DAN is used with BERT in Subtask A and 70.49 in Subtask B. Both these scores are a significant improvements from those obtained with GloVe representations (Table 1).

We retain the same configuration of BERT embedding layer for other experiments also. Training is performed with Adam (Kingma and Ba, 2015) optimizer with a learning rate of $1e{-}3$ for all the models.

3.4 CNN Classifier

The CNN classifier is composed of four 1-D convolution layers with filter widths ranging from two to five. Each convolutional layer has 192 filters. The output from each layer is max-pooled over sequence (time) dimension. This results in four 192d vectors, which are concatenated to get a 768d output.

The max-pooled outputs are passed through four fully connected feed forward layers with hidden dimensions of 768, 324, 162, and 2 respectively. The intermediate layers use ReLu activation and the final layer is a softmax layer. We use dropout of 0.2 on all layers of the feed forward network except for the final layer.

Without tri-training, this model obtains an absolute improvement of $\approx 4\%$ F_1-score over DAN in Subtask A. However in Subtask B, it performs poorer than the baseline DAN model with an F_1-score of 64.31. This decrease in performance could be because of overfitting on the source domain due to the larger number of parameters in CNN compared to DAN.

3.5 Tri-training

The aim of doing tri-training is for domain adaptation by labelling unseen data from a newer domain. For Subtask B, the CNN + BERT model achieves an F_1-score of 82.19 when trained with the tri-training procedure mentioned in Algorithm 1. Tri-training is used to label the 824 unlabelled sentences from the test set of Subtask B and augmented with the original training data. This score is a huge improvement from the classifier model trained only on the given data which gets an F_1-score of 64.31.

We also do the same experiment using 5000 unlabelled sentences from Yelp hotel reviews dataset (Blomo et al., 2013). The model obtains a similar score of 81.98 which proves the importance of tri-training in domain adaptation.

For Subtask A, we get an improvement in the F_1-score using tri-training, however the increase is not as profound as we observe for Subtask B. We compare the statistical significance of the different models and experiments in Section 3.7.

3.6 Upsampling

We also wanted to find how the class balance in the dataset has affected our model performance. The class distribution of the datasets including the test set distribution that was obtained after the final evaluation phase are mentioned in Table 2.

Dataset	Suggestions (%)
Training	23
Subtask A validation	50
Subtask B validation	50
Subtask A Test	**10**
Subtask B Test	42

Table 2: Label distribution

The original training data has a class imbalance with only 23% of the sentences labelled as suggestions. We tried to balance the labels by naive upsampling, ie., adding duplicates of sentences that are labelled as suggestions. This allowed us to have a balanced training dataset for our experiments. This resulted in consistent gains over the original dataset during the trial evaluation phase.

However during the final submission, in Subtask A we found that the model's performance in the test set did not correlate well with that of the validation set as shown in Table 1. This could be because the percentage of positive labels in the test set is only 10% while the validation set has 50%.

Experiments without upsampling gives better performance in test set even though there is a decrease in the validation score. For Subtask B however, upsampling has actually increased the model performance. On hindsight, this could be because of similar distribution of class labels in both validation and test sets.

The submitted models received an F_1-score of 68.07 in Subtask A and 81.03 in Subtask B.

3.7 Statistical Significance Test

Reichart et al. (2018) suggests methods to measure whether two models have statistically significant differences in their predictions on a single dataset. We incorporate a non-parametric testing method for significance called the McNemar's test recommended by them for binary classification. Pairwise comparison of few of our models are reported in Table 3. The table contains the p-values for the null hypothesis. The null hypothesis is that two models do not have significant differences in their label predictions. In simpler words, a small p-value for an experiment pair denotes a significant difference in the prediction disagreement between two models. For example, from Table 3, DAN + GloVe and DAN + BERT models have a p-value less than 0.05 in both sub-tasks. This indicates that there is significant disagreement between the predictions of two models. Since DAN + BERT gets a better F_1-score and $p < 0.05$, we can confidently assert that improvement is not obtained by chance.

We use majority voting from five random seeds to get the final predictions on the test set for doing the paired significance testing.

4 Conclusion

We discussed our experiments for doing suggestion mining using tri-training. Tri-training combined with BERT representations proved to be an effective technique for doing semi-supervised learning especially in a cross-domain setting. Future work could explore more optimal ways of doing tri-training, evaluate the effect of contextual representations in tri-training convergence, and try more sophisticated architectures for classification that may include different attention mechanisms.

Subtask	Model A	Model B	p-value
A	DAN +glove	DAN +bert	≈ 0
A	DAN +bert	CNN +bert	0.046
A	CNN +bert	CNN +bert +tritrain$_{Test}$	**0.108**
B	DAN +glove	DAN +bert	$1.419e-05$
B	DAN +bert	CNN +bert	**0.4208**
B	CNN +bert	CNN +bert +tritrain$_{Test}$	$3.251e-08$
B	CNN +bert +tritrain$_{Test}$	CNN +bert +tritrain$_{Yelp}$	**0.5862**

Table 3: Pairwise comparison of various models using the McNemar's Test. $p \leq 0.05$ indicates a significant disagreement between the model predictions.

References

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*.

Jim Blomo, Martin Ester, and Marty Field. 2013. Recsys challenge 2013. In *RecSys*.

Avrim Blum and Tom M. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT*.

Minmin Chen, Kilian Q. Weinberger, and John Blitzer. 2011. Co-training for domain adaptation. In *NIPS*.

Xilun Chen and Claire Cardie. 2018. Multinomial adversarial networks for multi-domain text classification. In *NAACL-HLT*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640.

Mohit Iyyer, Varun Manjunatha, Jordan L. Boyd-Graber, and Hal Daumé. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Roi Reichart, Rotem Dror, Gili Baumer, and Segev Shlomov. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *ACL*.

Sebastian Ruder and Jeremy Howard. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Sebastian Ruder and Barbara Plank. 2018. Strong baselines for neural semi-supervised learning under domain shift. In *ACL*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17:1529–1541.

ZQM at SemEval-2019 Task9: A Single Layer CNN Based on Pre-trained Model for Suggestion Mining

Qimin Zhou, Zhengxin Zhang, Hao Wu*, Linmao Wang
School of Information Science and Engineering, Yunnan University
Chenggong Campus, Kunming, P.R. China
{zqmynu,zzxynu}@gmail.com, haowu@ynu.edu.cn, wlmdyx@gmail.com

Abstract

This paper describes our system that competed at SemEval 2019 Task 9 - SubTask A: ”Suggestion Mining from Online Reviews and Forums”. Our system fuses the convolutional neural network and the latest BERT model to conduct suggestion mining. In our system, the input of convolutional neural network is the embedding vectors which are drawn from the pre-trained BERT model. And to enhance the effectiveness of the whole system, the pre-trained BERT model is fine-tuned by provided datasets before the procedure of embedding vectors extraction. Empirical results show the effectiveness of our model which obtained $9th$ position out of 34 teams with F1 score equals to 0.715.

1 Introduction

Suggestion mining is defined as the extraction of suggestions from unstructured text (Negi et al., 2018). Suggestion mining is still a relatively young research area as compared to other natural language processing issues like sentiment analysis (Negi and Buitelaar, 2015). While suggestion mining is of great commercial value for organisations to improve the quality of their entities by considering the positive and negative opinions collected from platforms. The target of this task is to automatically classify the sentences collected from online reviews and forums into two classes which are suggestion and non-suggestion respectively (Negi et al., 2019).

BERT which stands for **B**idirectional **E**ncoder **R**epresentation from **T**ransformers is the latest breakthrough in the field of NLP provided by Google Research (Devlin et al., 2018). It has substantially advanced the state-of-the-art in many NLP tasks, especially in question answering (Alberti et al., 2019). More importantly, it provides a widely applicable tool for representation learning which can be generalized to many NLP tasks.

For this subtask, we firstly learn the word or sentence embeddings utilizing the pre-trained BERT model. Then the embedding vectors are extracted from BERT as the input of the subsequent model. It is worth noting that we have fine-tuned the pre-trained BERT model with provided dataset before the embedding vectors are extracted. In other words, this part is equivalent to the conventional embedding layer. This strategy is a little bit like ELMO (Peters et al., 2018). As for the upper layer of this system, convolutional neural network (CNN) is adopted herein to process the features. Although CNN is originally invented for tackling computer vision issues, while it has subsequently been shown to be effective for many NLP tasks (Kim, 2014; Zhang and Wallace, 2015; Dong et al., 2015).

The remainder of the paper is organized as follows. Section 2 describes the detailed architecture of our system. Section 3 reports the experimental results on the given datasets. Finally, conclusions are drawn in Section 4.

2 System Description

Figure 1 gives a high-level overview of our approach. And we elaborate the details of implementation which mainly consists of following steps: (1) the preprocessing of raw data, (2) the word embedding learning via BERT model, (3) feature processing via CNN and sentences classification.

2.1 Data Preprocessing

The provided dataset is collected from feedback posts on Universal Windows Platform and annotated by (Negi et al., 2018). But the text is not standard enough as there are some spelling mistakes and few duplicate samples. To boost the performance of our system, we conduct some pre-

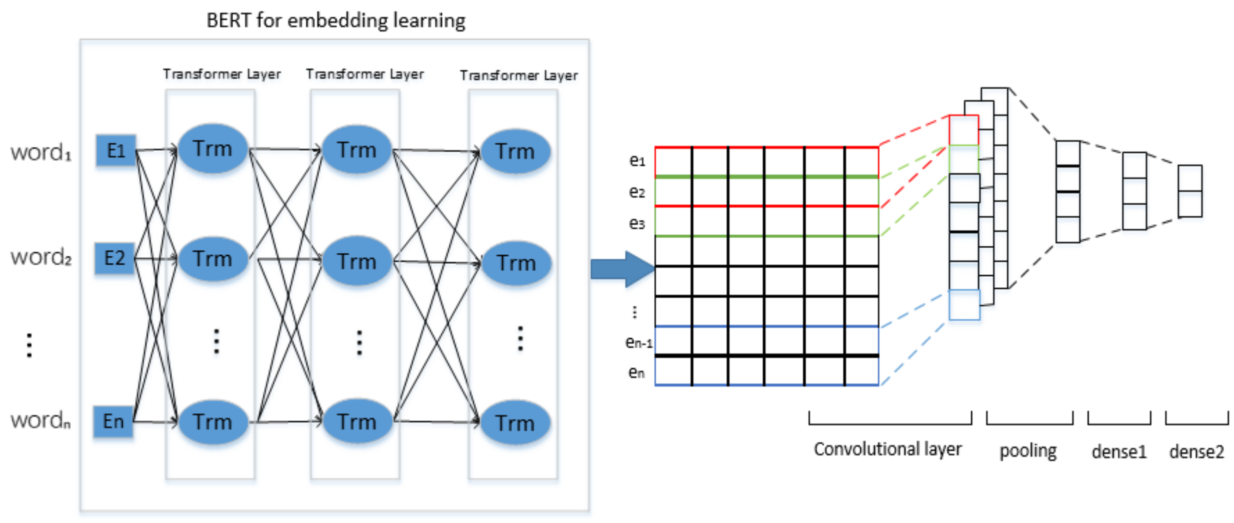


Figure 1: The architecture of our proposed model.

processing steps on the raw data. At first, web links are removed through regular expression as it does not contribute to the accuracy of classification. After that, we can take more meaningful words into consideration under the condition of a finite sentence length. And *ekphrasis* [1], a text processing tool, is utilized to conduct spelling correction (Baziotis et al., 2017). Then, all characters are converted to lowercase. Finally, duplicated samples would be excluded from the dataset.

2.2 Embedding Learning via BERT

Embedding layer usually encodes each word into a fixed-length vector for subsequent study. *Word2vec*, *Glove* and *FastText* are the most simple and popular word embedding algorithms (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2017). While there continue to be some drawbacks, such as they cannot encode the contextual information well. Recently, a few effective algorithms have been put forward such as *ELMO* and *openAI GPT* (Peters et al., 2018; Radford et al., 2018). These two pre-trained language models can encode rich syntactic and semantic information and distinguish the different meanings of a polysemy in diverse contexts, which traditional word embeddings methods cannot handle well. *ELMO* leverages the concatenation of independently trained left-to-right and right-to-left LSTM to generate features. Though LSTM can capture contextual information, the performance can be limited by the long distance of sequences to some extent. In order to deal with this problem, *openAI GPT* substitute *Transformer* for LSTM. *Transformer* rely entirely on attention mechanism to capture global dependencies (Vaswani et al., 2017).

BERT is the latest language representation model which also takes advantage of *Transformer*. Besides that, it uses the masked language model (MLM) and the bidirectional *Transformers* to capture the contextual information which has been proved to be effective (Devlin et al., 2018). In our system, we employ *BERT* as the embedding layer, in other words, we use the output of the last transformer layer from BERT as word embedding vectors. The version of pre-trained BERT model we used is BERT-BASE which has 12 layers of transformer blocks and the hidden size is equal to 768. It implies that the dimension of the output embedding vectors is equal to 768. We choose not to cover too much details of BERT as it has been elaborated on its website [2].

To boost the performance of our system by making this model better fit our data, a fine-tuning step is conducted before extracting the word embedding vectors from pre-trained model. And the fine-tuning parameters are given in Section 3.2.

2.3 Feature processing via CNN

CNN is originally invented for tackling the issues in the field of computer vision, while various C-

[1] github.com/cbaziotis/ekphrasis

[2] https://github.com/google-research/bert

NN models have subsequently been proven to be effective for many NLP tasks (Kalchbrenner et al., 2014; dos Santos and Gatti, 2014). In our work, we train a single layer CNN on the word embedding vectors drawn from BERT model. Let $e_i \in \mathbb{R}^k$ be the k-dimensional word embedding vector corresponding to the i-th word in the sentence. Then the vector representation of a sentence is denoted as Eq.1:

$$e_{1:n} = [e_1 \parallel e_2 \parallel e_3 \parallel ... \parallel e_n], \tag{1}$$

where n represents the length of sentences, and $\parallel$ denotes concat operation which can maintain the order of words in text. After that a filter involved in one-dimensional convolution operation is defined as $\mathbf{w} \in \mathbb{R}^{m \times k}$, then the convolution process can be defined as a function as Eq.2 (Kim, 2014):

$$p_i = f(\mathbf{w} \cdot e_{i:i+m-1} + b), \tag{2}$$

where p_i is a scalar which stands for the new local feature generated by a filter from a window of words $e_{i:i+m-1}$, in other words, only i-th to $i+m-1$-th words have been taken into consideration when generate i-th local feature. m is filter size which denotes m words is taken into calculation when generating a local feature. b is a bias and f is an activation function, it is $tanh$ exactly in our system. Finally, there are $n-m+1$ local features generated by a filter totally. Those local features can be concatenated as a global feature P:

$$P = [p_1 \parallel p_2 \parallel p_3 \parallel ... \parallel p_{n-m+1}], \tag{3}$$

where $P \in \mathbb{R}^{n-m+1}$ represents a feature map generated by a filter. Then a *max-over-time maxpooling* operation (Collobert et al., 2011) is applied to the feature maps which means that only the maximum value of P is reserved. If there are N_f filters, then N_f maximum values is generated through *maxpooling* operations. Those values can be organized as a new vector $Q \in \mathbb{R}^{N_f}$ as Eq.4:

$$Q = [max(P_1) \parallel max(P_2) \parallel ... \parallel max(P_{N_f})], \tag{4}$$

2.4 Dense layers

The pooling layer is followed by two dense layers with different number of neurons. *Dropout* (Hinton et al., 2012) is utilized to alleviate overfitting problem before the first dense layer. And we have tried different dropout rates to search the best configuration. Firstly, the output of pooling layer Q is fed into the first dense layer with 200 hidden neurons. The activation function of this dense layer is *relu* (Xu et al., 2015). Next is the second dense layer with two neurons and the corresponding activation function is *softmax*. Final output is the probability of which class the sample belongs to.

3 Experiments

3.1 Dataset

Classes	Train set	Trial Test set	Test set
0 (non-suggestions)	6415	296	746
1 (suggestions)	2085	296	87

Table 1: Data distribution

The available dataset released by organizer is split into three parts: train set, trial test set and test set. The positive and negative sample distribution of each part is described as Table 1. Apparently, there is class imbalance that the number of negative samples overwhelms the number of positive samples both in training set and test set. So for experiments, we fuse the train set and trail test set into a larger training set, and then split 10% samples as validation set randomly (8183 samples for training and 909 samples for validation). We train our model on the train set, tune the model parameters on the validation set, evaluate the model performance on the test set.

Model	Dropout	Macro average F1
Baseline	-	0.2676
Word2vec+CNN	0.4	0.3789
our model	0.1	0.7459
	0.2	0.7236
	0.3	0.7407
	0.4	0.7309
	0.5	0.7179
	0.6	0.7368
	0.7	0.7029

Table 2: The performance comparison.

Classes	Precision	Recall	F1 score
0 (non-suggestions)	0.98	0.96	0.97
1 (suggestions)	0.70	0.79	0.75

Table 3: The classification accuracy of different classes.

3.2 Experiment Results

As mentioned in Section 2.2, we conduct fine-tuning operation before extracting the word embedding vector from *BERT*. For fine-tuning, most hyperparameters are the same as the parameters

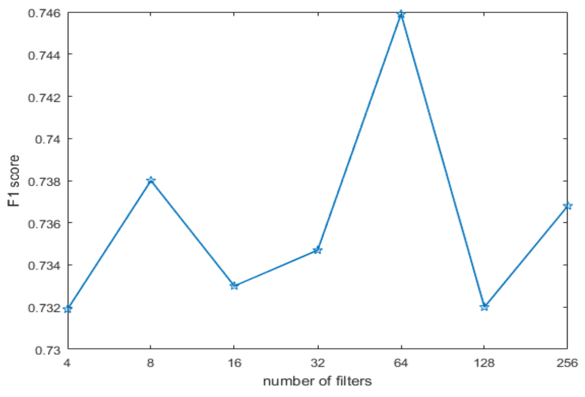


Figure 2: The impact of the number of filters.

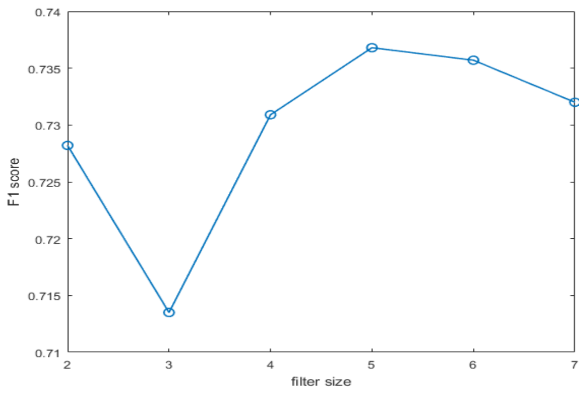


Figure 3: The impact of filter size.

of pre-trained model, with the exception of batch size, learning rate and number of training epochs. The mini-batch size is set at 32 and learning rate is set at 5e-5, the number of training epochs is configured as 5. The maximal length of sentences is configured as 50 and if the length of a sentence is less than 50, it will be padded with zero; otherwise, it will be truncated from the tail. For the CNN component, the filter size m is configured as 5 and the number of filters N_f is 64. And the dropout rates we have tried are ranged from 0.1 to 0.7 with a step of 0.1. The experimental results are shown in Table 2. In order to prove the effectiveness of word embeddings derived from BERT, we also employ Word2vec as comparison and the corresponding best dropout rate is 0.4. Obviously, no matter what the dropout rate is, our model consistently outperforms other models. And the best dropout rate of our model is around 0.1 for abovementioned configuration. While the best dropout rate may vary with other parameter configuration like filter size and the number of filters.

Table 3 shows the Precision, Recall and F1 score in term of different classes. Obviously, the system performance on negative samples is better than the performance on positive samples which is consistent with our intuition for the reason that the number of negative samples overwhelms the number of positive samples. Therefore, our system can learn more features of negative samples, which can help it recognize those negative samples accurately.

We also investigate the impact of the number of filters N_f on the classification performance with fixing the filter size m at 5 and the dropout rate at 0.1. The experimental result is shown as Figure 2. This model yields the best performance when the number of filters is equal to 64. Less filters cannot capture enough information while too many filters can result in information redundancy to some extent.

The impact of filter size on classification accuracy is shown as Figure 3. The most suitable filter size which means the window size of convolutional operation is $m = 5$. It is difficult for this system to catch the global semantic information if the window size is too small. While some local semantic information would be ignored if the window size of filter is too large. Hence, choosing a filter with moderate size is helpful for the performance improvement.

4 Conclusions

In this paper, we have proposed a neural model based on the pre-trained BERT model to deal with suggestion mining task. Our system can learn the representation of sentences or words effectively by leveraging BERT. Then the representations extracted from BERT is fed into a simple CNN layer. Experimental results show that our system is efficient on the given dataset.

As for future work, it is of necessity to tackle the imbalance between positive samples and negative samples through oversampling or undersampling. And we intend to study some innovative ways to incorporate BERT model like extracting not only the output of the last transformer layer but also the output of different transformer layers and integrating them with different weights.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (61562090).

References

Chris Alberti, Kenton Lee, and Michael Collins. 2019. A bert baseline for the natural questions. *CoRR*, abs/1901.08634.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 260–269.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *ACL*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Sapna Negi and Paul Buitelaar. 2015. Towards the extraction of customer-to-customer suggestions from reviews. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2159–2167.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Sapna Negi, Maarten de Rijke, and Paul Buitelaar. 2018. Open domain suggestion mining: Problem definition and datasets. *arXiv preprint arXiv:1806.02179*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

ProblemSolver at SemEval-2019 Task 10: Sequence-to-Sequence Learning and Expression Trees

Xuefeng Luo, Alina Baranova, Jonas Biegert
Linguistics Department, University of Tuebingen, Germany
firstname.lastname@student.uni-tuebingen.de

Abstract

This paper describes our participation in SemEval-2019 shared task "Math Question Answering", where the aim is to create a program that could solve the Math SAT questions automatically as accurately as possible. We went with a dual-pronged approach, building a Sequence-to-Sequence Neural Network pre-trained with augmented data that could answer all categories of questions and a Tree system, which can only answer a certain type of questions. The systems did not perform well on the entire test data given in the task, but did decently on the questions they were actually capable of answering. The Sequence-to-Sequence Neural Network model managed to get slightly better than our baseline of guessing "A" for every question, while the Tree system additionally improved the results.

1 Introduction

The data set for the task (Hopkins et al., 2019) includes questions used in the Math SAT. There are three broad categories of questions: closed-vocabulary and open-vocabulary algebra questions, and geometry questions. All types of questions consist in large part of natural language. Closed-vocabulary algebra questions typically contain math equations and many math-specific words, while open-vocabulary algebra questions include more everyday vocabulary and quantities which are expressed in letters, numbers or a combination of both. Geometry questions are usually provided with diagrams the analysis of which is necessary for solving the problems. Most questions of all categories are multiple-choice questions with five possible options, but some questions have a numeric answer.

We present two systems to tackle these math problems. One of them, a sequence-to-sequence LSTM model pre-trained with augmented data, is applied to all three types of questions, while the other, a system based on expression trees produces answers exclusively for open-vocabulary algebra questions.

2 Related Work

In their work, Roy and Roth (2015) introduced binary *expression trees* that represent and solve math word problems. To choose the best expression tree out of all possible trees, the authors employed two classifiers: a relevance classifier, which determined if the quantity should be included into the expression tree, and a Lowest Common Ancestor (LCA) classifier, which output the most probable mathematical operation for a pair of quantities. Both classifiers were trained on gold annotations.

Subsequently, two other systems were developed based on Roy and Roth (2015). One of the systems belongs to the same authors and uses the concept of Unit Dependency Graphs (UDGs) to capture the information between units of the quantities (Roy and Roth, 2017). UDGs are then united with expression trees, allowing the information about dependencies between units improve the math problem solver.

Another system (Wang et al., 2018) suggests a method to improve Roy and Roth's approach. By applying deep reinforcement learning, which has proved to be suitable for problems with big search space, the authors achieve better accuracy and efficiency.

An earlier system introduced by Wang et al. (2017) used gated recurrent units (GRU, Chung et al., 2014) and long short-memory (LSTM, Hochreiter and Schmidhuber, 1997) to automatically solve simple math word problems by converting words into math equations.

3 Model Description

3.1 Sequence-to-Sequence Neural Network

Our model is based on a sample implementation provided by the Keras team (Chollet et al., 2015). This model was able to calculate addition, such as from “535+61” to “596” with a Sequence-to-Sequence model using LSTM (Hochreiter and Schmidhuber, 1997). Similar to this model, our model also had 128 hidden units and started with an embedding layer with an alphabet of 96 characters. The longest question was 650 characters. Then, we used a LSTM as encoder. For all dighits in the answers, we have seperate vectors representing them. Thus, we have repeated vectors of outputs 5 time, in order to represent 5 dighits. Our decoder was another LSTM layer with returing sequences, followed by a time distributed layer of dense layers where activation function was softmax. In addition to this, we added a 0.2-rate Dropout layer (Srivastava et al., 2014) after embeding layer, encoder LSTM and decoder LSTM, to prevent over-fitting. On top of that, we found that reversing and doubling the inputs can greatly improve training performance, according to Zaremba and Sutskever (2015). The seq2seq model is shown by Figure 1. We did not encoded answers along with questions. We only compared the answers strings to the questions' choices and made our decisions.

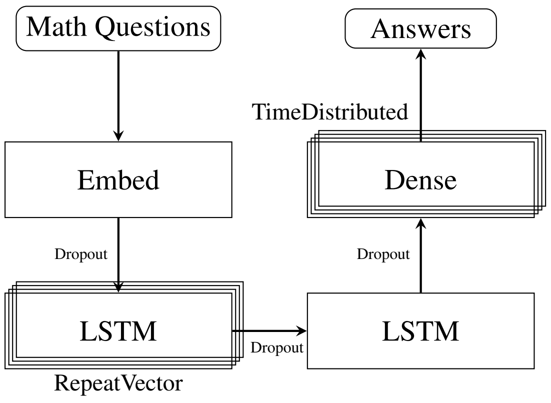


Figure 1: Seq2seq Model

We padded all answers into same length with extra space characters, but our model still was not able to produce exact answers with sequence-to-sequence. However, the sequences the model produced were good enough to predict the correct answer for multiple choice questions. For instance, for question “If $x+345 = 111$, what is the value of x?”, the output of the system would be “-234444”, which is very close to the correct answer “-234”. Thus, we wrote a program which was able to compare the initial characters (including “-”) regardless the extra characters at the end with answer options and predict the correct answer.

3.2 Tree System

The system of Roy and Roth (2015) has a lot of advantages: for instance, it can solve math problems that require multiple steps and different operations, and it can handle problems even if it did not see similar problems in the training set. That is why we chose to implement this approach for solving open-vocabulary algebra questions.

The expression trees the authors used in their system have a special structure that allows to calculate them in a simple and unambiguous way. In such a tree, the leaves are quantities extracted from the problem text, and the internal nodes are mathematical operations between the quantities. By calculating values of all internal nodes, one can obtain the value of the tree route, which corresponds to the answer of the problem.

Similarly to Roy and Roth (2015), we used the relevance and the LCA classifiers to evaluate all possible expression trees and choose the one that answers the problem correctly. However, instead of using gold annotations, we decided to train the classifiers on all the trees that result in right answers, partly because annotations were not available, and partly because we were curious how well the system could perform with no manual effort invested in annotating training data.

Tree evaluation was done by two simple multilayer perceptrons. As described earlier, the first one returns the probability of a given quantity to be relevant, as in a tree that answers the question correctly contains that quantity, the second one returns the probabilities for each of the possible operations to be the lowest common ancestor of a pair of given quantities in a tree that answers the question correctly.

For every possible tree per question, the product of the probabilities of each quantity to be relevant was added to the product of the probabilities of the lowest common ancestor of each quantity pair being correct. These scores, as well as the result of the tree were put in a list and ordered by score. The results of the trees were then matched against the answer options of each question and the answer option that was first matched in the list was given as the answer to the question. If the question had no answer options, the result of the highest rated tree was given as the answer to the question.

4 Experimental Settings

4.1 Sequence-to-Sequence Neural Network

4.1.1 Data Augmentation

Initially, we tried to trained our model directly on the questions, but it turned out that model could not learn at all. In total, we had slightly more than 1000 SAT questions, which was insufficient for an RNN model. Not to mention that the small training set contained questions with a variety of types – open- and closed-vocabulary algebra questions as well as geometry questions, leaving an even smaller training set for each subtype. Thus, data augmentation was a necessary step. In order to strengthen the connection of numbers, we did not provide original SAT data with numbers modified, but more than 600,000 simple closed-vocabulary algebra questions.

Among them, there were two types of questions augmented for our model. These included two types of questions within 3 digits like "If $x + 345 = 111$, what is the value of x?" and "If $x - 345 = 111$, what is the value of x?". Not only numbers and variable names were randomized but the positions of variables were switched. In toal, there were 614,236 questions, where "plus" had 330,620 and "minus" had 283,616 questions. Even though augmented data had large differences with SAT questions, results showed they still prodigiously contributed to our training.

4.1.2 Training

Rather than training our model together with original SAT data and augmented data, we chose to trained with augmented data first, and then continued to train with original data. There were 40 iterations of 614,236 questions dealing with addition and subtraction. Fractions were also present in the training set. After training with the augmented questions set, our model was trained with actual questions from the Math SAT. In total, there were 200 iterations of 805 Math SAT Questions. Nevertheless, since the training data was so small, it is highly possible that our model was prone to over-fitting to the training data.

Example 1
On a certain map, 100 miles is represented by 1 inch. What is the number of miles represented by 2.4 inches on this map?

4.2 Tree System

4.2.1 Quantities

Quantities were extracted from questions and answers using a rule-based approach. Before the extraction, all mentions of quantities were normalized to digits (e.g. *one* to *1*). Then, numbers, number-word combinations (e.g. *13-inch*), small letters denoting quantities (all letters except *a*) and LaTeX expressions were retrieved. LaTeX expressions that contained only numbers were transformed into numbers (e.g. \frac{*1*}{*10*} into *0.1*). In general, all questions that contained quantities other than numbers or the answer of which had several quantities were filtered out, leaving us with 75% of open-vocabulary questions from the training set. In the next stage, while constructing trees for the training data, we heuristically set the maximum number of quantities in a question to 7, which led to using 59% of the training data.

4.2.2 Operations and Tree Enumeration

Once quantities from the question were extracted, all their possible combinations were obtained, with size from two to the total number of quantities. The order of quantities in these combinations, however, stayed the same. Consider Example 1. For this word problem, the combination [100 2.4] would be possible, but the combination [2.4 100] would not.

For every combination obtained in the previous step, all possible expression trees with quantities as leaves and empty inner nodes were generated. These inner nodes were filled with all possible combinations of operation signs. As in earlier studies (Roy and Roth, 2017; Wang et al., 2018), we used six operations: apart from the standard $+$, $-$, $\times$ and $\div$, we included reverse operators $-_{\text{rev}}$ and $\div_{\text{rev}}$ to account for the fact that the order of quantities stays the same in their combinations.

Like Roy and Roth (2015), we implemented constraints that define *monotonic* trees. These constraints are concerned with the order of multiplication operator in regard to division operator, and the order of addition operator in relation to subtraction operator. However, unlike the authors, we used these constraints to decrease the number

System	Accuracy
Baseline (always "A")	14.3%
Baseline + seq2seq	15.0%
Baseline + trees	15.9%
Baseline + seq2seq + trees	16.7%

Table 1: Results

of trees resulting in right answers, not to guarantee that any monotonic tree for the solution expression has the same LCA operation for any pair of quantities in it, as in Roy and Roth (2015).

4.2.3 Features

We used UDPipe (Straka and Straková, 2017) to parse questions' text and extract features for the classifiers. The features are identical to the ones that Roy and Roth (2015) describe.

5 Results

The results that our systems achieved are shown in Table 1. Our official submission consists only of the neural network, which achieved 15%, with the accuracy on closed-vocabulary algebra questions being 16% and the accuracy on the other two categories being 15% each. This result, however, was achieved by guessing "A" whenever the question could not be answer by the model. When guessing is removed, the overall accuracy drops to 2%. However, on the 109 questions the model could actually answer, it achieved 21% accuracy.

In post-evaluation, after combining the results of the neural network and the tree system, we were able to achieve 17% accuracy overall by increasing the accuracy on open-vocabulary algebra questions by 20%. If we remove the guessing, the tree system achieves 3% accuracy overall, which stems from its 13% accuracy on open-vocabulary algebra questions. If we only count the questions that could actually be answered by the system, its accuracy would be equal to 26%. Without guessing, the combination of both systems produces 4% accuracy overall, with the distribution being 2% on closed-vocabulary algebra questions, 13% on open-vocabulary algebra questions and 0.4% on geometry questions. On the 205 questions answered by the combination of both systems, the accuracy was 23%.

6 Discussion/Conclusion

The results of our systems on the full data set are, frankly put, rather poor. Nevertheless, the tree system shows promising results in solving open-vocabulary questions, if it is refined and improved, while the neural network seems not to perform well on any specific type of questions, although its overall performance is similar to that of the tree system.

Concerning the neural network, it might be beneficial to focuse on specific types of questions, instead of trying to train a model that deals with mixed types of questions. RNN best learnt on closed- and open-vocabulary algebra questions, therefore training separate models for these types could be one way to improve the system. In addition to that, a much larger dataset is critical in enhancing the model, thus promoting the accuracy of its predictions. Lastly, data augmentation would further improve the model. If we were to train a versatile model for mixed types of math questions, we could perform data augmentation on each type.

The current problem of the tree system lies to a large extent within the quality of the tree evaluation. It heavily relies on answer options to be available, as the average index of the first tree that produces an answer option in the score list is 47 (for the test data). Therefore, the answer of the highest-rated tree would most likely be wrong. Other aspects that could be improved include choosing other features for the classifiers, decreasing the scores of trees with low amounts of quantities (those trees are currently overrated) or using a different machine learning algorithm altogether, such as deep reinforcement learning (e.g Wang et al., 2018).

Apart from that, using no additional quantities in constructing trees, and including every quantity once made it difficult to obtain trees that not only gave the right result for the questions from the training set, but also answered them in a right way. Moreover, expanding expression trees to problems that involve letters to denote quantities would definitely contribute to improving the performance of the tree system.

7 Acknowledgements

Part of the experiments reported on this paper was run on a Titan Xp donated by the NVIDIA Corporation.

References

François Chollet et al. 2015. Keras. `https://keras.io`.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Mark Hopkins, Ronan Le Bras, Cristian Petrescu-Prahova, Gabriel Stanovsky, Hannaneh Hajishirzi, and Rik Koncel-Kedziorski. 2019. Semeval-2019 task 10: Math question answering. In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, USA.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2017. Unit dependency graph and its application to arithmetic word problem solving. In *AAAI*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. 2018. Mathdqn: Solving arithmetic word problems via deep reinforcement learning. In *AAAI*.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854.

Wojciech Zaremba and Ilya Sutskever. 2015. Learning to execute. *Computing Research Repository*, arXiv:1410.4615. Version 3.

RGCL-WLV at SemEval-2019 Task 12: Toponym Detection

Alistair Plum[1] [*], **Tharindu Ranasinghe**[1] [*], **Pablo Calleja**[2], **Constantin Orăsan**[1], **Ruslan Mitkov**[1]
[1]Research Group in Computational Linguistics, University of Wolverhampton, UK
[2]Ontology Engineering Group, Universidad Politécnica de Madrid, ES
`{a.j.plum, t.d.ranasinghehettiarachchige}@wlv.ac.uk`
`{c.orasan, r.mitkov}@wlv.ac.uk`
`pcalleja@fi.upm.es`

Abstract

This article describes the system submitted by the RGCL-WLV team to the *SemEval 2019 Task 12: Toponym resolution in scientific papers*. The system detects toponyms using a bootstrapped machine learning (ML) approach which classifies names identified using gazetteers extracted from the GeoNames geographical database. The paper evaluates the performance of several ML classifiers, as well as how the gazetteers influence the accuracy of the system. Several runs were submitted. The highest precision achieved for one of the submissions was 89%, albeit it at a relatively low recall of 49%.

1 Introduction

Resolving a toponym, a proper name that refers to a real existing location, is a non-trivial task closely related to named entity recognition (NER) (Piskorski and Yangarber, 2013). For this reason, using an NER system to detect and assign location tags could seem a good way forward. However, NER systems may not be able to detect whether a name refers to a actual location or not (e.g., *London* in *London Bus Company*). In addition, location names are usually ambiguous, which means it is crucial that these are disambiguated in order to assign the correct coordinates.

While in the past the focus in toponym resolution has been on rule and gazetteer driven methods (Speriosu and Baldridge, 2013), more recent approaches also consider ML-based techniques. DeLozier et al. (2015) describe their ML-based approach, which does not require a gazetteer. The approach calculates the geographical profile of each word, which is refined using Wikipedia statistics, and then fed into an ML classifier. Speriosu and Baldridge (2013) also make use of an ML classifier which is text-driven. Geotags of documents are used to automatically generate a training set. Although the two previous approaches used two standard corpora for toponym resolution, consisting of news articles and 19th century civil war texts, there are wide areas of application for toponym resolution. For instance, Ireson and Ciravegna (2010) explore the use in social media, while Lieberman and Samet (2012) attempt to analyse news streams. Spitz et al. (2016) have also used an encyclopaedic dataset, compiled from Wikipedia, WordNet and GeoNames.

The focus of the SemEval 2019 Task 12 was toponym resolution in journal articles from the biomedical domain (Weissenbacher et al., 2019). The articles that had to be processed were case studies on the epidemiology of viruses, meaning that the developed systems can potentially be used to track viruses. The task was composed of three sub-tasks: (1) toponym detection, followed by (2) disambiguation and the assignment of the appropriate coordinates, as well as (3) the development of an end-to-end system.

This paper presents our participation in the first sub-task. Our system performs first a gazetteer look-up for locations, and then uses machine learning (ML) to classify whether or not it represents an actual location. The gazetteers are extracted from the online geographical database GeoNames, whilst the classification is carried out by feeding the context of potential locations in an ML classifier. The rest of the paper presents the system developed (Section 2), followed by its evaluation (Section 3). The paper finishes with conclusions.

2 System Description

The system developed for this task was designed as a pipeline consisting of three stages: text clean-

[*]The first two authors contributed equally to the paper.

ing, text processing and identification of locations. The rest of this section presents each of these stages. The system has been made available on online. [1]

2.1 Text Cleaning

The first processing stage identifies parts of the text which do not contain any locations that have to be identified according to the task guidelines. These parts include the references section of each text and the information about authors of the journal articles. In addition, the texts also contain genome sequences and abbreviations of chemicals, which resemble abbreviations for locations. Regular expressions were used to replace these text sequences with spaces. We chose to replace the sequences rather than remove them in order to keep the correct offsets of entities which are crucial in the evaluation process.

Not all the genome sequences were correctly identified due to the variability of how they are represented. As a result, not all these sequences were being replaced by spaces. This introduced noise in our processing pipeline. In addition, in some cases the regular expressions for excluding the references section would fail to correctly identify the boundaries of this section. Since this left large amounts of these texts blank, three texts did not have their respective references sections removed.

2.2 Text Processing

Once cleaned, the texts were processed using components from the ANNIE pipeline within GATE (Cunningham et al., 2002, 2011). The ANNIE pipeline was designed for named entity recognition tasks, but for our purpose we used only the tokeniser and gazetteer lookup components.

We produced three different gazetteers. The first one contained all locations from the GeoNames geographical database. The second gazetteer contained a list of cities from GeoNames with a population of over 5,000. The third gazetteer features a list of countries, capitals, and cities with a population larger than 15,000 people extracted from GeoNames. The default list of regions included with ANNIE was also used. A list of US regions as well as their abbreviated forms was added manually.

[1]https://github.com/TharinduDR/SemEval-2019-Task-12-Toponym-Resolution-in-Scientific-Papers

The output of this module was a list of annotations, including tokens boundaries and tokens matching the gazetteer entries. This information is then used by the ML classifier in the next step.

2.3 Identification of Locations

Once the texts are processed, the next task is to detect whether a candidate location really refers to a location. In addition to cases of common nouns which may also be used as a location, there are also cases where the location names were used as adjectives. For example, in the sentence *Other mutations observed in the HA gene of the Kentucky isolates have also been reported by others*, even though the gazetteer identifies *Kentucky* as a location it is actually referring to a virus entity. According to the guidelines, this should not be annotated as a location, making the task quite difficult.

Analysis of examples from the training data indicated that the context of candidate locations can be used to assess whether the detected word is an actual location or not. For this reason, we trained a machine learning model which uses the context of candidates to distinguish between real locations and falsely identified locations by the gazetteer look-up component. For the experiments presented here, we used a window of two words before and two words after the candidate location to obtain its context. More precisely, if the detected word from the gazetteer is ω_i, the context c_i was defined as,

$$c_i = \omega_{i-2} + \omega_{i-1} + \omega_i + \omega_{i+1} + \omega_{i+2} \qquad (1)$$

The annotated gold standard provided by the task organisers was used to create a training set which contained both positive and negative instances. Two machine learning approaches were considered for this word window classification task. The first approach was to use traditional machine learning models, while the other approach was to use neural network models.

2.3.1 Traditional ML Approach

There are multiple ways that words can be translated into a numerical representation before they can be used as features for a machine learning model. The commonly used representations convert sequences of words to a bag of words or tf-idf vectors. However, since their introduction, word embedding models (Mikolov et al., 2013) have been widely used as features for text classification tasks and have proven successful. In addition, they have the capability to represent the context better than tf-idf vectors. For this reason, we used the 300 dimensional word2vec embedding model trained on the Google news corpus.

The word windows had to be represented by a vector that can be fed as features to a machine learning model, while retaining a unique length over all the training and testing examples, in order to be input into a traditional machine learning model. There are many ways to represent a text window with word embeddings. Simply averaging the word embeddings of all words excluding stop words in a text has proven to be a strong baseline or feature across a multitude of tasks, such as short text similarity tasks (Kenter et al., 2016). Following that, the mean of word vectors in a particular word window was calculated in order to represent the whole word window with a vector, which is a 300 dimensional vector in this scenario. The vector calculated was used as features and fed into several machine learning classifiers such as Support Vector Machines (Cortes and Vapnik, 1995), Random Forest classifier (Breiman, 2001) and XGBoost (Chen and Guestrin, 2015). The parameters were tuned using 10-fold cross validation. For the implementation scikit-learn in python 3.6 was used.

2.3.2 Neural Network Architectures

The representation described above performs poorly on classification tasks such as sentiment analysis, because it loses the word order in the same way it happens with the standard bag-of-words model, and fails to recognise many sophisticated linguistic phenomena (Le and Mikolov, 2014). For this reason, the second approach relies on neural networks which receive as input the embedding vectors corresponding to the context, but without performing any modification on it. Keras was used to implement these neural architectures.

Two neural architectures were developed. The first one was adopted from text classification research (Coates and Bollegala, 2018). As depicted in figure 1 it contains variants of Long Short-Term Memories (LSTMs) with self attention followed by average pooling and max pooling layers. It also has a dropout (Srivastava et al., 2014) between 2 dense layers after the concatenate layer. The model was trained with cyclical learning rate (Smith, 2017).

The pooling layers in the first architecture are considered as a very primitive type of routing

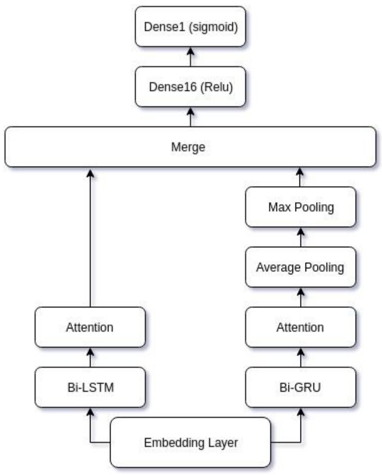


Figure 1: LSTM variant with self attention

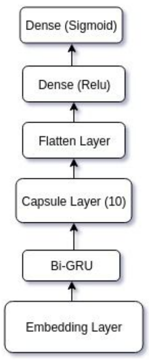


Figure 2: Capsule net architecture

mechanism. The solution that is proposed is a capsule network (Sabour et al., 2017). A capsule network with a bi-directional GRU was also experimented with for this data set. The complete architecture is shown in figure 2. There is a spatial drop out (Tompson et al., 2015) between the embedding layer and bi-directional GRU layer and there is also a dropout (Srivastava et al., 2014) between two dense layers after the capsule layer.

The results and evaluation criteria of both traditional approaches and neural network approaches are reported in the results section 3.

3 Results

3.1 Gazetteers

As described in the previous section, three different gazetteers were tested using the development and training sets. As the machine learning component of the system would make the final prediction, it was important to ensure the maximum number of candidate locations. Therefore, it was

Gazetteer	Precision	Recall	F-Score
GN all	0.2359	**0.7699**	0.3612
GN 5000+	**0.3584**	0.7563	0.4863
GN custom	**0.3546**	**0.7678**	0.4851

Table 1: Gazetteer evaluation results

vital to ensure the highest possible recall, while achieving acceptable precision results.

Table 1 shows the precision, recall and F-score values for each of the gazetteers, described in section 2, run on the training set. Rows one and two had a high recall but low precision, and a higher precision, but lower recall, respectively. Row three shows the results for the final gazetteer. It has the best balance between precision and recall, and was selected for use in the final system.

3.2 Identification of Locations

Locations in the training set were matched using the gazetteers and then extracted together with their respective word window, in order to compile a separate data set. This data was split into a training set and an evaluation set for the machine learning classifiers. The training set consisted of 80% of the total data set and the evaluation set, containing the gold standard annotations from the previous training set, had the rest of the 20%. The accuracy of each machine learning model evaluated on the evaluation set is shown in Table 2. Predictions were considered to be accurate if the machine learning model and the gold standard matched, including correct and incorrect classifications. All other cases were considered to be non-accurate.

Our baseline - a zero-R classifier predicting every instance as a falsely identified location had an accuracy of 71.95%. All of our machine learning models were able to outperform the baseline model significantly, even though the data set is in-balanced. The capsule net architecture, which provided the best performance at an accuracy of 88.73%, was selected for use in the final system.

3.3 Submission Results

After we had determined the best components for the system, the GN custom gazetteer and the bi-GRU + Capsule architecture, the whole system was evaluated on the test set. The submission results are presented in four categories, determined by the organisers. Table 3 shows the results for

ML Model	Accuracy
Zero-R	71.95%
Random Forest	84.21%
SVM	83.44%
XGBoost	85.80%
Bi-LSTM/Bi-GRU + Max Pooling	87.75%
Bi-GRU + Capsule	**88.73%**

Table 2: ML models evaluation results

Test	Precision	Recall	F-Score
Strict macro	0.8280	0.4746	0.6034
Strict micro	0.8168	0.3396	0.4798
Overlap macro	**0.8980**	**0.4969**	**0.6398**
Overlap micro	0.8936	0.3654	0.5187

Table 3: Final submission results

each. Overall, our system achieves the highest values in the overlap macro class, with the lowest in the strict micro class. The system tends to achieve acceptable precision scores, but at low recall values. This trend can most probably be explained by the fact that many candidate locations are not detected by the gazetteers. Together with the machine learning part discarding some proper locations, this has a dramatic affect on the recall.

4 Conclusion and Future Work

This paper presented the system we submitted to the *SemEval 2019 Task 12: Toponym resolution in scientific papers*. Evaluation of the system has shown that a pipeline that combines traditional string matching and advanced machine learning can offer promising results. It has demonstrated that a larger size of the gazetteer does not necessarily have a positive effect on performance. It has also made clear that a higher recall value for the gazetteer look-up component could provide a much better basis on which to train machine learning approaches. On the machine learning side, we have demonstrated that employing word embeddings together with state-of-the-art algorithms can be a viable way of classifying toponyms.

Due to time constraints, a large amount of different parameters, as well as optimizing the lookup algorithm and underlying gazetteers were not tested or carried out. For future research we hope to address these problems, so that a better basis on which to train machine learning architectures can be achieved, as well as more deep learning architectures.

References

Leo Breiman. 2001. Random forests. *Machine Learning*, 45:5–32.

Tianqi Chen and Carlos Guestrin. 2015. Xgboost : Reliable large-scale tree boosting system.

Joshua Coates and Danushka Bollegala. 2018. Frustratingly easy meta-embedding - computing meta-embeddings by averaging source word embeddings. In *NAACL-HLT*.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20:273–297.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Niraj Aswani, Ian Roberts, Genevieve Gorrell, Adam Funk, Angus Roberts, Danica Damljanovic, Thomas Heitz, Mark A. Greenwood, Horacio Saggion, Johann Petrak, Yaoyong Li, and Wim Peters. 2011. *Text Processing with GATE (Version 6)*.

Grant DeLozier, Jason Baldridge, and Loretta London. 2015. Gazetteer-independent toponym resolution using geographic word profiles. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Neil Ireson and Fabio Ciravegna. 2010. Toponym resolution in social media. In *The Semantic Web – ISWC 2010*, pages 370–385, Berlin, Heidelberg. Springer Berlin Heidelberg.

Tom Kenter, Alexey Borisov, and Maarten de Rijke. 2016. Siamese cbow: Optimizing word embeddings for sentence representations. *CoRR*, abs/1606.04640.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*.

Michael D. Lieberman and Hanan Samet. 2012. Adaptive context features for toponym resolution in streaming news. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 731–740, New York, NY, USA. ACM.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Jakub Piskorski and Roman Yangarber. 2013. Information extraction: Past, present and future. In *Multi-source, multilingual information extraction and summarization*, pages 23–49. Springer.

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. 2017. Dynamic routing between capsules. *CoRR*, abs/1710.09829.

Leslie N. Smith. 2017. Cyclical learning rates for training neural networks. *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472.

Michael Speriosu and Jason Baldridge. 2013. Text-driven toponym resolution using indirect supervision. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1466–1476.

Andreas Spitz, Johanna Geiß, and Michael Gertz. 2016. So far away and yet so close: Augmenting toponym disambiguation and similarity with text-based networks. In *Proceedings of the Third International ACM SIGMOD Workshop on Managing and Mining Enriched Geo-Spatial Data*, GeoRich '16, pages 2:1–2:6, New York, NY, USA. ACM.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. 2015. Efficient object localization using convolutional networks. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 648–656.

Davy Weissenbacher, Arjun Magge, Karen O'Connor, Matthew Scotch, and Graciela Gonzalez. 2019. Semeval-2019 task 12: Toponym resolution in scientific papers. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

THU_NGN at SemEval-2019 Task 12: Toponym Detection and Disambiguation on Scientific Papers

Tao Qi , Suyu Ge , Chuhan Wu , Yubo Chen , Yongfeng Huang
Tsinghua National Laboratory for Information Science and Technology,
Department of Electronic Engineering, Tsinghua University Beijing 100084, China
{qit16,gesy17,wuch15,chen-yb18,yfhuang}@mails.tsinghua.edu.cn

Abstract

Toponym resolution is an important and challenging task in the neural language processing field, and has wide applications such as emergency response and social media geographical event analysis. Toponym resolution can be roughly divided into two independent steps, i.e., toponym detection and toponym disambiguation. In order to facilitate the study on toponym resolution, the SemEval 2019 task 12 is proposed, which contains three subtasks, i.e., toponym detection, toponym disambiguation and toponym resolution. In this paper, we introduce our system that participated in the SemEval 2019 task 12. For toponym detection, in our approach we use TagLM as the basic model, and explore the use of various features in this task, such as word embeddings extracted from pre-trained language models, POS tags and lexical features extracted from dictionaries. For toponym disambiguation, we propose a heuristics rule-based method using toponym frequency and population. Our systems achieved 83.03% strict macro F1, 74.50 strict micro F1, 85.92 overlap macro F1 and 78.47 overlap micro F1 in toponym detection subtask.

1 Introduction

Toponym resolution is an important task in the natural language processing field and has many applications such as emergency response and social media geographical event analysis(Gritta et al., 2018). Toponym resolution is usually modelled as a two-step task. The first step is toponym detection, which is a typical named entity recognition (NER) task. The second step is toponym disambiguation, which aims to map locations to its coordinates in the real world.

NER is a widely explored task and most NER methods can be applied to toponym detection. For example, Ratinov and Roth (2009) used n-grams, history predictions as the input features of conditional random fields (CRF) for toponym detection. Usually the performance of these methods heavily relies on the quality of hand-crafted features. However, manually selected features may be sub-optimal. Also, these methods cannot effectively exploit contextual information due to the dependency on bag-of-word features. In recent years, many neural network based methods have been proposed for NER. For example, Ma and Hovy (2016) proposed a CNN-LSTM-CRF model for NER. They use CNN layer to learn character features of each word, LSTM layer to learn the contextual word representations and CRF layer to predict the label jointly. Gregoric et al. (2018) proposed Parallel RNN architecture. They split a single LSTM into multiple equally-size ones with a penalty to promote diversity. However, these methods cannot utilize external knowledge to recognize entities, which is usually important to toponym detection. Usually, linguistic knowledge such as part-of-speech and dictionary knowledge may be useful for toponym detection, and they are easy to obtain. Therefore, in this paper, we aim to incorporate these external knowledge sources to enhance our neural model for toponym detection.

Similarly, there are many works on toponym disambiguation. Most of them are rule-based methods. They use some heuristics to rank the candidates and choose the highest one(Gritta et al., 2018). For example, Karimzadeh et al. (2013) used the geographical level(e.g. country, province and city), the Levenshtein Distance and the population of potential candidates to rank the candidate toponym and choose the highest one. However, the result of toponym disambiguation relied on corpus domain and the rule should be reconsidered when applied to different corpus.

For the toponym detection task, we use TagLM(Peters et al., 2017) as the basic model.

In our model, we first learn word representations from original characters, then learn contextual word representations by a stacked Bi-LSTM network, and finally use a CRF layer to jointly decode the label sequence. To enrich the representations of words, we incorporate various features such as pre-trained word embeddings, POS tags and lexicon features. For the toponym disambiguation task, we design a rule-based heuristics method by using toponym frequency and population to rank candidate toponyms. Our systems achieved 83.03% strict macro F1 in the toponym detection task, 67.21% in the toponym disambiguation task and 61.31% strict macro F1 in toponym resolution.

2 Our Approach

2.1 Toponym Detection

Our model is based on TagLm (Peters et al., 2017). As shown in Fig. 1, our model have three major components, i.e., *character encoder*, *feature concatenation* and *toponym detector*.

Usually, character patterns are important clues for toponym detection. For example, starting with a capital letter (e.g. Eastern Europe), all cased word (e.g. UK) and mixed cased word (e.g. HongKong) are very common in toponym names. Thus, we use a *character encoder* module to learn word representations from original characters. There are two layers in the *character encoder*. The first one is a character embedding layer. It converts each character in a word into a low-dimensional dense vector. The second one is a character-level CNN layer. It was used to capture local contextual information. We also apply a max pooling layer to build word representations by selecting the most salient features.

The *feature concatenation* module is used to concatenate different types of features. There are four types of additional features in our model, i.e., pre-trained word embeddings, pre-trained language model word representations, POS tag representations and lexicon representations. Usually, word embeddings are pre-trained on a large corpus and can provide rich semantic information. Thus, we use pre-trained word embeddings to enrich word representations by incorporating semantic information . However, word embeddings usually do not contain contextual information. Thus, we also incorporate word representations generated by pre-trained language models. Usually, toponyms have specific POS tags such as nouns. Following Wu et al. (2018), we also incorporate POS tag information to guide our model. There are two layers in our model to learn POS tag representations. The first one is a POS tag embedding layer, which learns low-dimensional embedding vectors for POS tags. The second one is a Bi-GRU layer. It was used to learn the syntax structure of sentences and output the hidden POS tag representations. In addition, since many toponyms can be found in toponym databases, lexical features may be useful for toponym detection. Due to our observations, toponym names are more likely to have low occurrence frequency in documents and less number of matched toponyms in the toponym database. Thus we constructed three one-hot vectors as lexical features. First, we counted the number of matched toponyms in the database for every word. They were quantified to different levels and represented by the first one-hot vectors. The second one-hot vectors were used to represent whether the first matched toponym 's names returned from the database were perfect matches. The third one-hot vectors were used to represent quantified occurrence frequency in the training set of every word. Besides, a three-layer feed-forward neural network (FFNN) was used to learn lexical representations for every word as a lexical representation.

The *toponym detector* module aims to predict the label of each word from its representations. There are two submodule in the toponym detector. The first one is a stacked Bi-LSTM network. Usually, global contexts are important for toponym detection. For example, in the sentence "Beijing is the capital of China", the words "capital" and "China" are all informative for toponym detection. Thus, we use a stacked Bi-LSTM network to learn hidden word representations based on global contexts. The second one is a CRF layer, which is used to decode the label sequence jointly (Lafferty et al., 2001). Usually, there is relatedness between the labels of neighbor words. For example, the label "I" (inside) can only appears after "B" (beginning). Thus, we use CRF to do joint label decoding.

2.2 Toponym Disambiguation

Toponym disambiguation is a down-stream task of toponym detection. Due to the lack of dictionary knowledge, it's difficult for a neural network to do toponym disambiguation. Thus, we propose

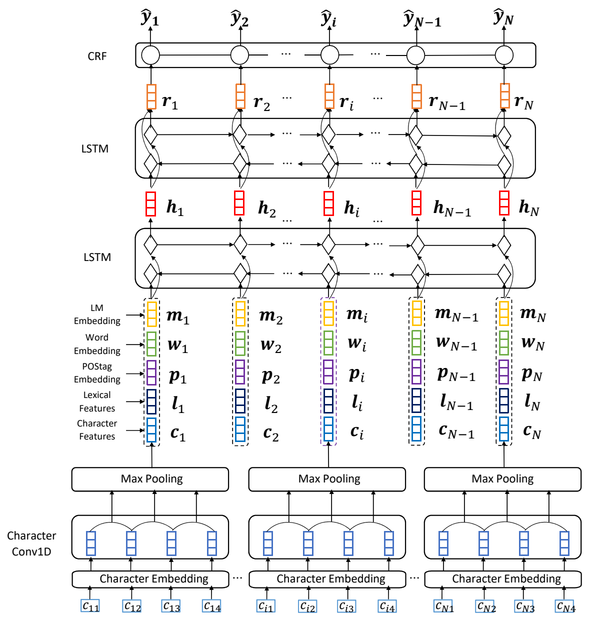


Figure 1: The architecture of our model for toponym detection.

a rule-based heuristic method for toponym disambiguation. An observation is that among the toponym candidates returned by the database, the toponym with higher frequency is more likely to be mentioned. In addition, the toponym with higher population may also have a higher probability to be mentioned. Therefore, we propose a heuristic algorithm named Most Frequency - Most Population (mFmP). If a toponym appears in the train set, we will select the highest frequency id as the output. Otherwise, we will select the toponym with the most population as output.

3 Experiment

3.1 Experimental Settings

We conduct experiments on science reports provided the SemEval-2019 task 12. The data set is composed of 72 full-text journal articles in open access. There are four different metrics to evaluate the prediction performance, i.e., strict macro F1, strict micro F1, overlap macro F1 and overlap micro F1.

In the toponym detection task, we used NLTK[1] for sentence segmentation, word tokenization and POS tagging. We used ELMo(Reimers and Gurevych, 2017) and BERT(Devlin et al., 2018) model to generate 1024-dimensional contextualized word embeddings. We used GeoNames[2] to construct lexical feature. The BIO tagging scheme(Sang and Veenstra, 1999) was used in the toponym detection task. In the toponym disambiguation task, we use GeoNames database to retrieve candidate toponyms.

In our approach, the three word embedding vectors we used (Glove(Pennington et al., 2014), word2vec(Mikolov et al., 2013), fasttext(Bojanowski et al., 2017)) were all 300-dimensional. The dimension of the character embedding was set to 100. The character CNN had 100 filters, and their window size was set to 3. The sizes of the 3-layer FFNN were respectively set to 256, 256, and 128. We set the dimension of POS tag embeddings to 128. The Bi-GRU layer for POS tag representation learning was 64-dimensional. The two Bi-LSTM layers for capturing long-distance and short-distance information were 128-dimensional and 64-dimensional. To mitigate overfitting, we added 20% dropout to each layer. We used Adam as the optimizer for model training.

In our approach, we used transductive learning techniques to further improve the performance of our approach. We first trained our model on the train set, and then applied our model to the test set to generate pseudo labeled data. Finally, we jointly trained our model on the combination of the training and test sets. In addition, we use model ensemble strategy to reduce the uncertainty of our model(Wu et al., 2017). We trained our model for 10 times independently and the final predictions are made by voting.

3.2 Performance Evaluation

In this section, we compare our approach with several baseline methods to evaluate the performance of our approach. The baseline methods are listed as follows. (1) **Baseline**: a baseline system provided by SemEval 2019 task 12(Davy et al., 2019). It uses n-grams as input and a FFNN network to predict label. The input features includes word embedding and character features. (2) **CNN-CRF**: a two-layer CNN and a CRF layer with word embedding for toponym detection. (3) **LSTM-CRF**: a two layer LSTM and a CRF layer with word embedding for toponym detection.

The comparative results are listed in Table 1. According to these experimental results, we have several observations. First, *LSTM-CRF* outperforms *CNN-CRF*. This may be because CNN can

[1]https://www.nltk.org
[2]http://www.geonames.org

method	SMA	SMI	OMA	OMI
baseline	75.56	69.84	80.55	82.60
CNN-CRF	51.28	42.20	61.23	52.51
LSTM-CRF	61.26	47.73	73.26	61.56
Our approach	**84.10**	**82.36**	**91.36**	**90.72**

Table 1: Performance of different toponym detection methods. SMA, SMI, OMA, and OMI respectively denote the strict macro F1, strict micro F1, overlap macro F1 and overlap micro F1.

method	strict macro F1	strict micro F1
baseline	**84.00**	77.59
mFmP	83.58	**78.14**

Table 2: Performance evaluation of toponym disambiguation.

only capture local information, instead, LSTM can utilize global information. This indicates that capturing global contextual information have the potential to improve the performance of toponym detection. Second, the baseline method outperforms *LSTM-CRF* and *CNN-CRF*. This may because *LSTM-CRF* and *CNN-CRF* did not use character-level features, which shows the effect of character-level information on the performance of toponym detection. In addition, the performances of *CNN-CRF* and *LSTM-CRF* are very poor. This may because these two models only use word embedding to enhance the model's semantic information. And this word embedding is not trained on science reports dataset, which may make these two methods lack of semantic information. Third, our approach outperformed all these baseline methods. This is because our approach use pre-trained word embeddings and language model to enhance the semantic information of model, use character-level word representations to capture character patterns of toponym names, and features, i.e., lexicon features and POStag features, to add extract information. This result validates the effectiveness of our model.

The results for toponym disambiguation are listed in Table 2. Baseline method selected toponym with highest population among candidate toponyms. The performance of our method is similar to baseline method. This may be because high frequency toponyms are often with large population.

Method	SMA	SMI	OMA	OMI
WE	77.50	62.61	83.82	69.33
WE+CE	81.75	66.16	85.78	70.16
TagLM	85.04	81.78	90.39	89.61
TagLM+POS	83.64	78.35	90.03	88.87
TagLM+LEX	**85.37**	77.77	90.91	86.57
TagLM+POS+LEX	84.10	**82.36**	**91.36**	**90.72**

Table 3: Influence of different features on the performance of our model. WE, CE, POS, LEX respectively denote toponym detector using word embedding, character encoder, POS tag representations and lexicon representations as input.

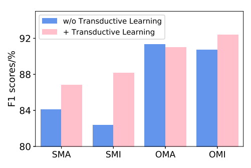
3.3 Influence of Different Features

In this section, we conduct several experiments to evaluate the effect of each type of features we used. We added different types of features to our toponym detector gradually to conduct our experiments. The experimental results are listed in Table. 3. According to Table. 3, we have several observations.

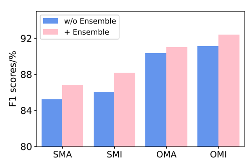
First, the WE+CE method consistently outperforms WE. This indicates that character-level word representations can make our model detect toponym names effectively. Second, the performance of TagLM is much better than the performances of WE and WE+CE. This may because WE and WE+CE method can only obtain semantic information from word embedding and training data, which is not enough. Incorporating word embedding vectors generated by pre-trained language models could enhance the semantic information of our model. Third, after incorporating POS tag embedding or lexical feature into our model separately, the performances of these two models declined. This may be because the POS tag and lexical features of several samples are inaccurate, which incorporate misleading information into our model. Forth, incorporating these two features into our model together improve the performance of our model. This may be because both features have inherent relatedness and our model is more easily to exploit useful information from the combination of both features.

3.4 Influence of Transductive Learning and Model Ensemble

In this section, we conduct several experiments to evaluate the influence of transductive learning and model ensemble on the performance of our model. The experimental results are shown in Figure. 2.



(a) Influence of transductive learning.



(b) Influence of model ensemble

Figure 2: Influence of transductive learning and model ensemble.

According to Figure. 2, we can find both transductive learning and ensemble strategy can improve the performance of our model. This indicates that our model could be more robust by incorporating more training samples and voting scheme.

4 Conclusion

In this paper, we introduce our system participating in the SemEval-2019 task 12. For the toponym detection, we use a TagLM model with various features to enrich word representations. In addition, we use a transductive learning method and ensemble strategy to further improve the performance of our model. For toponym disambiguation, we propose a heuristics rule-based method based on toponym frequency and population. Our systems achieve 83.03% strict macro F1 in toponym detection, 67.21% strict macro F1 in toponym disambiguation and 61.31% strict macro F1 in toponym resolution.

Acknowledgments

This work was supported by the National Key Research and Development Program of China under Grant number 2018YFC1604002, and the National Natural Science Foundation of China under Grant numbers U1836204, U1705261, U1636113, U1536201, and U1536207.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Weissenbacher Davy, Magge Arjun, O'Connor Karen, Scotch Matthew, and Gonzalez Graciela. 2019. Semeval-2019 task 12: Toponym resolution in scientific papers. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Andrej Zukov Gregoric, Yoram Bachrach, and Sam Coope. 2018. Named entity recognition with parallel recurrent neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 69–74.

Milan Gritta, Mohammad Taher Pilehvar, Nut Limsopatham, and Nigel Collier. 2018. Whats missing in geographical parsing? *Language Resources and Evaluation*, 52(2):603–623.

Morteza Karimzadeh, Wenyi Huang, Siddhartha Banerjee, Jan Oliver Wallgrün, Frank Hardisty, Scott Pezanowski, Prasenjit Mitra, and Alan M MacEachren. 2013. Geotxt: a web api to leverage place references in text. In *Proceedings of the 7th workshop on geographic information retrieval*, pages 72–73. ACM.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 338–348, Copenhagen, Denmark.

Erik F Sang and Jorn Veenstra. 1999. Representing text chunks. In *Proceedings of the ninth conference on*

European chapter of the Association for Computational Linguistics, pages 173–179. Association for Computational Linguistics.

Chuhan Wu, Fangzhao Wu, Yongfeng Huang, Sixing Wu, and Zhigang Yuan. 2017. Thu_ngn at ijcnlp-2017 task 2: Dimensional sentiment analysis for chinese phrases with deep lstm. *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 47–52.

Chuhan Wu, Fangzhao Wu, Sixing Wu, Zhigang Yuan, and Yongfeng Huang. 2018. A hybrid unsupervised method for aspect term and opinion target extraction. *Knowledge-Based Systems*, 148:66–73.

UNH at SemEval-2019 Task 12: Toponym Resolution in Scientific Papers

Matthew Magnusson
Department of Computer Science
University of New Hampshire
mfm2@cs.unh.edu

Laura Dietz
Department of Computer Science
University of New Hampshire
dietz@cs.unh.edu

Abstract

The SemEval-2019 Task 12 is toponym resolution in scientific papers. We focus on Subtask 1: Toponym Detection which is the identification of spans of text for place names mentioned in a document. We propose two methods: 1) sliding window convolutional neural network using ELMo embeddings (CNN-ELMo), and 2) sliding window multi-Layer perceptron using ELMo embeddings (MLP-ELMo). We also submit a bi-directional LSTM with Conditional Random Fields (bi-LSTM) as a strong baseline given its state-of-art performance in Named Entity Recognition (NER) task. Our best performing model is CNN-ELMo with a F1 of 0.844 which was below bi-LSTM F1 of 0.862 when evaluated on overlap macro detection. Eight teams participated in this subtask with a total of 21 submissions.

1 Introduction

Toponyms are textual spans of text identifying geospatial locations. This can range from the canonical name of populated places, such as "London" to direct or indirect mentions of geographic entities. The parsing of geographic locations from unstructured text is considered an open challenge due to domain diversity, place name ambiguity, metonymic language and often limited leveraging of context (Gritta et al., 2018). Many scientific publications contain toponyms which can be challenging to extract automatically. Specifically, names of institutions and viruses often contain geographic references which may confuse the extractor. Often, the extractor needs to handle noisy text parsed from PDF versions of scientific articles which can introduce artifacts.

In Task 12, a toponym is defined to include proper names and geographic entities but to exclude indirect mentions of places and metonyms. Additional discussion of the motivation and task description is available at the task web site.[1]

2 Related Work

There is significant work in the area of toponym detection (Matsuda et al., 2015; D. Lieberman et al., 2010) and the closely related fields of named entity recognition (NER) (Li et al., 2018) and entity mention detection (EMD) (Shen et al., 2015) with many different approaches. State-of-the-art named entity detection models have historically employed a combination of hand-crafted features, rules, natural language processing (NLP), string-pattern matching, and domain knowledge using supervised learning on manually annotated corpora (Piskorski and Yangarber, 2018). A common approach to toponym detection has been to utilize place name gazetteers which are directories of geographic names and their corresponding geolocations to perform string matching of place names in text (D. Lieberman et al., 2010).

Contemporary approaches in entity detection have utilized neural-based architectures. (Collobert et al., 2011) propose a window-based, multi-layer, dense feed-forward neural architecture using word embeddings concatenated with orthographic features and a gazetteer as an input layer with a hard Tanh output layer for superior performance on a standard NER task. Huang et al. (2015) utilise a bi-directional LSTM with a sequential conditional random layer using a gazetteer and Senna word embeddings to obtain superior performance. Magge et al. (2018) achieves state-of-the-art results in toponym detection by utilizing a window-based deep neural network, word embeddings trained on a domain-specific corpus, orthographic features, and a gazetteer.

[1]https://competitions.codalab.org/competitions/19948

Table 1: Gold Standard Corpus Statistics

	Documents	Tokens	Toponyms
Train	72	396,668	3,637
Valid	32	179,443	2,141
Test	45	253,159	4,616
Total	149	829,720	10,394

3 Data

A gold standard corpus, composed of 150 full text journal articles in open access from PubMed Central (PMC), is provided by the task organizers.[2] Additional information can be found at Weissenbacher et al. (2017) for the general approach followed by the task organizers for developing the corpus. Table 1 highlights the gold standard corpus statistics.

4 Approach

Our approach is motivated by the simplicity and strong performance of windows-based approaches on the NER and toponym task with the strong performance of deep contextual embeddings on related NLP tasks. Two different neural based approaches are undertaken by the team: 1) sliding windows convolutional neural network using deep embeddings, and 2) sliding window multi-layer perceptron using deep embeddings. The embeddings are composed of an ELMo contextual embedding concatenated with hand-crafted features.

4.1 Embeddings

The ELMo embeddings (Gardner et al., 2018) are learned functions of the internal states of a deep bidirectional language model (biLM) that has been pre-trained on the 1B Word Benchmark. These vectors are developed from the concatenation of each of the 1,024 length vector outputs from the model for each token and are a function of the complete input sentence.

For each token in the context of its sentence, a vector representation is generated by concatenating the ELMo model embedding with the one-hot encoding of orthographic features and an additional flag bit indicating if the token was contained within the set of gazetteer tokens. This resulted in a vector of length 3,081 for each token. A padding vector of all 0s was used for the sliding window neural models.

[2]We are unable to successfully parse one of the documents from the train set due to an encoding error

4.2 Hand-crafted Features

Hand-crafted features were added as they slightly improve model performance when compared to using the ELMo embeddings alone for the input layer to the neural models.

Orthographic Features: a one hot encoding is assigned to each token based on its orthographic structure: only numeric, all lower case alphabetic characters, all upper case alphabetic characters, title-case alphabetic characters, mixed case (not title-case) alphabetic characters, alphabetic characters with numeric, padding token, and the "other" for the remaining tokens not matched by previously listed features. Alphabetic characters are UTF-8.

Gazeteer Features: a set of toponynm tokens is generated from the entries in GeoNames.[3] For example, for the entry in Geonames,"Gulf of Mexico", the tokens "Gulf", "of", and "Mexico" are added to the toponym set. This is used as a binary feature for the presence of the parsed token in the constructed Geonames token set.

4.3 Implementation details

The documents are parsed into sentences and tokenized using the open-source NLP library Spacy.[4]

Pre-trained embeddings are provided by Pyysalo et al. (2013).[5] which are generated from Wikipedia, PubMed, and PMC texts using the word2vec tool. They are 200-dimensional vectors trained using the skip-gram model with a window size of 5, hierarchical softmax training, and a frequent word subsampling threshold of 0.001. These vectors are used in the baseline and the bi-LSTM with CRF models.

ELMo embeddings are generated using the AllenNLP tool .[6] The deep learning library Keras 2.2.1 [7] is used for training the neural models. Tensorflow 1.12[8] is the backend used for training and evaluating all of the models attempted. In training the models, the Adam optimizer in Keras is used. Additional code and data will be available in an on-line appendix.[9]

[3]https://www.geonames.org/export/
[4]https://spacy.io/
[5]http://bio.nlplab.org/
[6]https://github.com/allenai/allennlp/
[7]https://keras.io/
[8]https://www.tensorflow.org/
[9] https://cs.unh.edu/ mfm2/index.html

4.4 Models

We compare the following models:

MLP-ELMo: A sliding window (size = 5) is applied to each sentence with padding vectors applied to boundary tokens. The input layer to the neural models is a 5 x 3081 matrix using the ELMo-based embeddings. The input layer is connected to two fully connected layers with 128 hidden units each and relu activation. The output is a sigmoid with a binary output to indicate if the token is part of a toponym.

CNN-ELMo: A sliding window (size = 5) is applied to each sentence with padding vectors applied to boundary tokens. The input layer is a 5 x 3081 layer. The input layer is two 1d convolutional layers with filter sizes of 250 and a kernel size of 3. A global 1-d max pooling layer follows the convolutional layers. Two fully connected layers with 100 hidden units each and relu activation follow max pooling. A sigmoid function is applied in output layer to indicate if the token is part of a toponym.

4.5 Baseline

Two models are used for evaluation: 1) a sliding window mlp provided by the task organizers, and 2) bi-LSTM with CRF. The bi-LSTM with CRF model demonstrates state-of-the-art results on NER and is used as an additional strong benchmark for model comparison.

MLP-Baseline: The task organizers provide a state-of-the-art geoparser as a strong baseline. The system has a specific component for toponym detection using a two-layer feedforward neural network (200 hidden units per layer) as described in Magge et al. (2018). The baseline features a sliding window (size = 5) over each sentence using Wikipedia-Pubmed-PMC word2vec embeddings for token encoding. The baseline did not include a gazetter-based lookup but did incorporate orthographic structure of the tokens: 1) All Caps - ASCII, 2) First letter capitalized - ASCII, and 3) first letter not-capitalized - ASCII. The baseline also uses separately trained vectors if the token contained a digit or unknown token in the vocabulary.

Bi-LSTM-Baseline: This strong baseline implementation utilizes the code developed by Reimers and Gurevych (2017).[10] Input sentences for the model are generated in the CoNLL format with the IOB representation for labeled toponyms in the training data. The embeddings are the Wikipedia-Pubmed-PMC word2vec vectors. Each LSTM has a size of 100 and is trained with a dropout of 0.25. Character embeddings are generated using a convolutional neural network and the maximum character length is 50. The model is fit over 25 training epochs.

[10]https://github.com/UKPLab/emnlp2017-bilstm-cnn-crf

5 Experiment Evaluation

5.1 Metrics

The metrics for evaluation are precision, recall and F-measure. Two variants are provided for toponym detection: strict and overlapping measures. In the strict measure, mentions match the gold standard if they match exact span boundaries. In the overlapping measure, a match occurs when the mention and gold standard share any common span of text.

There are two methods for computing precision and recall: micro averaging, and macro averaging. In micro averaging the corpus of documents is treated as one large document when calculating precision and recall. In macro averaging precision, recall and f-measure are calculated on a per document basis, and then the results are averaged.

5.2 Results

The model results are shown in Tables 2, 3, 4, and 5. The best model in the task by the Team "DM_NLP" (Davy Weissenbacher, 2019) is provided for comparison with the results our team achieves. The F1 scores of the two sliding window models and the bi-LSTM benchmark outperforms the task benchmark on all metrics. Each model is run once with a random model parameter initialization. The MLP-ELMo model had a similar feedforward structure and approach as the baseline neural model. The primary difference is the embedding vectors used. For both strict and overlap toponym detection, MLP-ELMo model achieves the same or higher precision and recall than the baseline.

The convolutional network using the ELMo-based embeddings exhibits higher performance on the f1 score relative to MLP-ELMo, however precision is higher and recall is lower for both strict and overlap measures.

The best performing model is the bi-LSTM with CRF method. This shows that the sliding window models with deep contextual embeddings did

Table 2: Overlap Macro

Run	P	R	F1
Bi-LSTM-Baseline	**0.910**	**0.819**	**0.862**
CNN-ELMo	0.908	0.788	0.844
MLP-ELMo	0.886	0.798	0.840
MLP-Baseline	0.864	0.797	0.829
DM_NLP	0.946	0.924	0.935

Table 3: Overlap Micro

Run	P	R	F1
Bi-LSTM-Baseline	0.897	**0.704**	**0.789**
CNN-ELMo	**0.913**	0.697	0.791
MLP-ELMo	0.890	0.737	0.807
MLP-Baseline	0.880	0.687	0.772
DM_NLP	0.954	0.880	0.915

Table 4: Strict Macro

Run	P	R	F1
Bi-LSTM-Baseline	**0.862**	**0.781**	**0.819**
CNN-ELMo	0.836	0.737	0.784
MLP-ELMo	0.811	0.740	0.774
MLP-Baseline	0.791	0.740	0.764
DM_NLP	0.927	0.906	0.916

Table 5: Strict Micro

Run	P	R	F1
Bi-LSTM-Baseline	**0.835**	**0.650**	**0.731**
CNN-ELMo	0.807	0.618	0.700
MLP-ELMo	0.782	0.646	0.707
MLP-Baseline	0.775	0.603	0.678
DM_NLP	0.929	0.856	0.891

not achieve state-of-art performance on this task. However, bi-LSTM with CRF has lower performance than the best model submitted for the task which indicates that other approaches can exceed the performance of a state-of-the-art Named Entity Recognition model.

Fine tuning shows that a window size of 5 yields the best performance on the validation set during training for the sliding window neural models. The sliding window neural models do not require many epochs of training with approximately only three required before overfitting of the training data becomes evident. Adding dropout to training did not appear to improve sliding window model performance.

6 Conclusion

The best performing submission by our team is bi-LSTM with CRF. This is not surprising as this technique has achieved state-of-the-art results in NER NLP tasks. The sliding window models we propose are similar in the approach as the task baseline model. The ELMo-based embeddings do achieve a boost in performance relative to baseline given the richer context and character structure they embed. This indicates that the ELMo-derived embeddings are superior in the task to embeddings trained on a domain-specific corpus using word2vec. However, for both overlap and strict macro the recall for MLP-ELMo is identical to the baseline model.

Bi-LSTM with CRF and the baseline neural model are noteworthy in that they are both able to extract toponym mentions only using context from embeddings to acheive high-quality results without relying on the presence of a gazetteer. An open question is if a gazetter or other knowledge graph structure could be incorporated into a deep neural model using contextual embeddings to achieve superior performance. It is also not clear why CNN-ELMo has lower recall than MLP-ELMo and baseline.

The results suggest that a sliding window model can be enhanced by better-quality embeddings and a convolutional component. The sliding window model approach is attractive due to its relatively straight-forward implementation and quick training time. The results achieved by bi-LSTM with CRF and the model submitted by DM_NLP, suggest that other approaches may ultimately generate superior performance on the toponym detection task.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1846017. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Michael D. Lieberman, Hanan Samet, and Jagan Sankaranarayanan. 2010. Geotagging: Using proximity, sibling, and prominence clues to understand comma groups.

Karen O'Connor Matthew Scotch Graciela Gonzalez Davy Weissenbacher, Arjun Magge. 2019. Semeval-2019 task 12: Toponym resolution in scientific papers. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *ACL workshop for NLP Open Source Software*.

Milan Gritta, Mohammad Taher Pilehvar, Nut Limsopatham, and Nigel Collier. 2018. What's missing in geographical parsing? *Lang. Resour. Eval.*, 52(2):603–623.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *CoRR*, abs/1508.01991.

Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2018. A survey on deep learning for named entity recognition. *CoRR*, abs/1812.09449.

Arjun Magge, Matthew Scotch, Abeed Sarker, Davy Weissenbacher, and Graciela Gonzalez-Hernandez. 2018. Deep neural networks and distant supervision for geographic location mention extraction. *Bioinformatics*, 34(13):i565–i573.

Koji Matsuda, Akira Sasaki, Naoaki Okazaki, and Kentaro Inui. 2015. Annotating geographical entities on microblog text. In *Proceedings of The 9th Linguistic Annotation Workshop*, pages 85–94. Association for Computational Linguistics.

Jakub Piskorski and Roman Yangarber. 2018. Chapter 2 information extraction : Past , present and future.

Sampo Pyysalo, Filip Ginter, Hans Moen, Tapio Salakoski, and Sophia Ananiadou. 2013. Distributional semantics resources for biomedical text processing.

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 338–348, Copenhagen, Denmark.

W. Shen, J. Wang, and J. Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge & Data Engineering*, 27(2):443–460.

Davy Weissenbacher, Abeed Sarker, Tasnia Tahsin, Matthew Scotch, and Graciela Gonzalez. 2017. Extracting geographic locations from the literature for virus phylogeography using supervised and distant supervision methods. *AMIA Jt Summits Transl Sci Proc*, 2017:114–122. 28815119[pmid].

UniMelb at SemEval-2019 Task 12: Multi-model Combination for Toponym Resolution

Haonan Li♠ Minghan Wang♠ Maria Vasardani♡
Martin Tomko♡ Timothy Baldwin♠
♠ School of Computing and Information Systems
♡ Department of Infrastructure Engineering
The University of Melbourne
{haonanl5,minghanw}@student.unimelb.edu.au,
{maria.vasardani, tomkom}@unimelb.edu.au, tb@ldwin.net

Abstract

This paper describes our submission to *SemEval-2019 Task 12 on toponym resolution in scientific papers*. We train separate NER models for toponym detection over text extracted from tables vs. text from the body of the paper, and train another auxiliary model to eliminate mis-detected toponyms. For toponym disambiguation, we use an SVM classifier with hand-engineered features. Our best model achieved a strict micro-F1 score of 80.92% and overlap micro-F1 score of 86.88% in the toponym detection subtask, ranking 2nd out of 8 teams on F1 score. For toponym disambiguation and end-to-end resolution, we officially ranked 2nd and 3rd, respectively.

1 Introduction

Toponym resolution (TR) refers to the task of automatically assigning geographic references to place names in text, which has applications in question answering and information retrieval tasks (Leidner, 2008; Daoud and Huang, 2013; Vasardani et al., 2013), user geolocation prediction (Roller et al., 2012; Han et al., 2014; Rahimi et al., 2015), and historical research (Grover et al., 2010).

This paper describes our system entry to the *Toponym resolution in scientific paper task of SemEval 2019* (Weissenbacher et al., 2019). The task consists of three subtasks: toponym detection, toponym disambiguation, and end-to-end toponym resolution.

For the toponym detection task, we extract tables from the full text and train separate BiLSTM-ATTN models for each. For tables, the model captures the horizontal row-wise structure of the table. For non-table content, the model can capture syntactic and semantic features. In both cases, we use a deep contextualized word representation — ELMo (Peters et al., 2018) — to represent each token. After detecting toponyms, we use an organization name detection model to eliminate mis-detected toponyms that are actually part of an organization name. For the toponym disambiguation task, we first construct a candidate set by searching toponyms on GeoNames.[1] Then, we manually construct features based on search results, and finally, train an SVM model to disambiguate the locations. For the end-to-end resolution task, we pipeline the two aforementioned steps.

Our work makes the following contributions:

- we show that training separate models for table and non-table portions of the paper is better than simply training one model over the full text;
- we show that contextualized word representation boosts performance;
- we show that auxiliary organization name recognition model is helpful for toponym detection, and better than training a single named entity recognizer (NER).

2 Toponym Detection

Figure 1 shows our workflow on the toponym detection task, which consist of 4 parts: (a) preprocessing, which contains tokenization, table extraction and sentence segmentation; (b) training and inference for the toponym detection model; (c) post-processing, to combine detected words into toponyms; and (d) refinement of the results by incorporating an auxiliary model.

2.1 Pre-processing

Tables are ubiquitous in scientific articles, and differ in structure to text in the body of the paper, in

[1]GeoNames, https://www.geonames.org/ is a freely available global placename database.

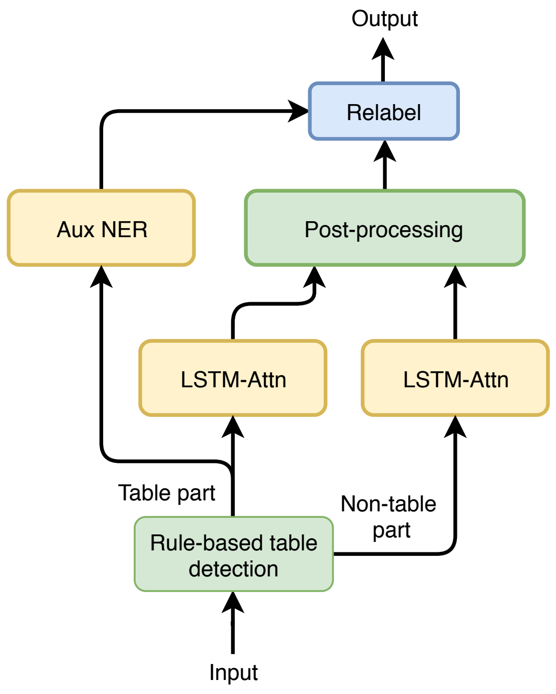


Figure 1: Toponym detection workflow

terms of syntactic structure. As such, training a single text embedding model over both the main body of text and tables will likely lead to sub-optimal representations, leading us to train separate models for: (1) tables, and (2) the remainder of the text content of the paper. To extract tables from the plain text dump provided by the shared task organisers, we use a rule-based table detection method.

We first tokenize the entire article, as part of which we treat all punctuation as a separator. In the process of table extraction, we process the raw text line-by-line rather than performing sentence tokenization. We treat numbers, OOV tokens (using GloVe vocabulary), |, and - as table elements, and consider lines with more than 70% of table elements to be table rows. Three or more consecutive table rows are considered to make up a table. In this way, we extract tables from the plain text dump of the articles. Note that the original PDF versions of papers were not made available by the task organizers, meaning that it wasn't possible to use vision-based methods to identify tables.

For the remainder of the text dump not detected as tables, we perform tokenization, remove hyphens caused by line breaks, and then perform sentence segmentation using SpaCy.[2] Sentences that are shorter than 5 tokens in length are concatenated with the preceding and proceeding sentences to make up a single sentence. By expanding short sentences, richer context can be exploited by both ELMo and the RNN-based model.

[2]https://spacy.io

2.2 Contextual Representation

We use ELMo (Peters et al., 2018) word representations in this paper, which are learned from the internal states of a deep bidirectional language model (biLM), pre-trained on a large text corpus. ELMo representations are purely character-based, allowing the network to use morphological clues to form robust representations for out-of-vocabulary tokens unseen in training. They are also robust to syntactic disfluencies caused by the fine-grainedness of word segmentation. For the purposes of empirical comparison, we also report on experiments using GloVe (Pennington et al., 2014) embeddings.

2.3 Models

For toponym extraction in the table part, we experimented with two kinds of models. The first is a token-level model which is described in Magge et al. (2018). In this model, each training instance consists of an input word, the word's context, and a label indicating whether the word is a part of a toponym. The context of the word is formed by the words in its neighbourhood, which is a window of words centred on the given word. We experimented with two- and three-layer feed-forward models.

The second model is built with RNN and self-attention (Vaswani et al., 2017). Although an RNN is able to make predictions over long sequences, the documents in this task are too long for an RNN, and at the same time, the size of the training data is not sufficient to train an RNN. As such, we split each document with several sentences and make predictions on separate sentences (hidden states are not passed through sentences). We use a two-layer bidirectional LSTM (Hochreiter and Schmidhuber, 1997) to capture the sequential information of the body and table contents, and use self-attention to enhance the connection of each token in the line. We consider the paper body content to have semantic information which can be captured by a sequential model like a BiLSTM. However, for tables, it is not clear that sequential information across cells in a table row should be processed as a sequence. Therefore, we use self-attention to learn the table structure over an entire line. We consider each sentence as a matrix which we denote as L, where $L \in \mathbb{R}^{t\times h}$; t represents the number of tokens in the line, and h is the dimensionality of the embedding representation. To improve training efficiency, we pack l lines into a single batch, thereby making L become a three-dimensional tensor $L \in \mathbb{R}^{l\times t\times h}$. We pad short lines to make the same length as the longest line in a batch, and set the embedding of each padding word to a zero vector with dimensionality h.

We first encode each line with a two-layer bidirectional LSTM, denoted as:

$$L' = \text{BiLSTM}(L) \tag{1}$$

Then, we feed L' into the attention model to encode structural information. The attention model can be denoted as follows:

$$\begin{aligned}&\text{Attn}(Q, K, V; \theta_Q, \theta_K, \theta_V) = \\ &\text{Softmax}\left(\frac{f(Q;\theta_Q)f(K;\theta_K)^\top}{\sqrt{h}}\right) f(V;\theta_V)\end{aligned} \tag{2}$$

This style of attention is named scaled dot-product attention by Vaswani et al. (2017), where $Q, K, V \in \mathbb{R}^{t\times h}$ represent the query, key, and value, respectively, and can be described as mapping a query and a set of key–value pairs to an output, where the query, keys, values, and output are all vectors. In this model, we use tokens in the same line to represent the query, key, and value, and use the attention function Attn to find self-correlations among them. Meanwhile, in Eqn 2 we define f to be a one-layer feed-forward network with different parameter sets $\theta_Q, \theta_K, \theta_V$, which we denote as $f(X;\theta)$. This allows us to learn the correlation with these three parameter sets. We use the following attention function:

$$L'' = \text{Attn}(L', L', L') \tag{3}$$

Finally, we pass L'' into a 3-layer feed-forward network denoted as g, using layer normalization in each layer to increase the training speed. The output of the feed-forward block is passed into the output layer with a residual connection with L'', denoted as:

$$\hat{y} = \phi(g(L'') + L'') \tag{4}$$

The architecture of the model is shown in Figure 2.

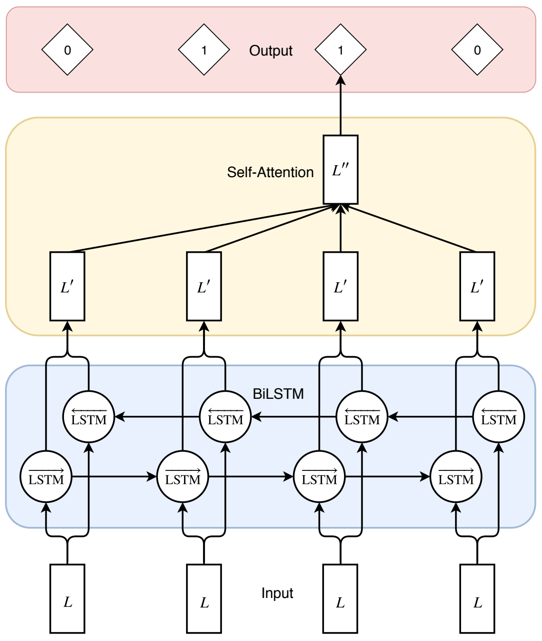


Figure 2: RNN with self attention

2.4 Post-processing

Since we are training a sequence labelling model, result segmentation and combination is necessary. For instance, the sentence *AIV H9N2 was spread to New York, Washington DC and Ottawa* contains three toponyms, and 5 tokens which are contained in those toponyms (e.g. the words *New* and *York* are combined into one toponym). An external gazetteer[3] downloaded from GeoNames, and an in-house place name abbreviation library were used.

We first restore all abbreviations in order to facilitate matching in the gazetteer. We then combine all consecutive tokens that were labelled as a toponym. After this, two different segmentation methods were used: (1) longest string match in the gazetteer; and (2) no segmentation. The result shows that the second method is better because of the limitation of string matching. We think using a better toponym match method like searching via Geonames rather than string matching could achieve better results.

2.5 Auxiliary Model

The single NER model picks up on features such as the word-initial character being uppercase, that are also common in non-toponym named entities, possibly resulting in toponym named entity FPs.

[3]http://download.geonames.org/export/dump/allCountries.zip

Model		Overlap Micro			Strict Micro		
		Precision	Recall	F1	Precision	Recall	F1
Table	BiLSTM	78.11	69.54	73.58	74.91	65.77	70.04
	BiLSTM+Attn	80.27	73.06	76.50	76.63	69.14	72.69
Non-table	BiLSTM	93.37	**90.69**	92.01	89.64	84.55	87.02
	BiLSTM+Attn	93.65	90.38	91.99	90.11	**84.78**	87.36
	BiLSTM+Attn+Aux	**94.66**	90.23	**92.39**	**90.98**	84.50	**87.62**
Single	BiLSTM+Attn+Aux	90.15	85.22	87.62	85.73	71.67	78.07
Combined	BiLSTM+Attn	90.77	86.27	88.46	85.02	72.59	78.31
Combined	BiLSTM+Attn+Aux	91.35	82.83	86.88	84.69	77.48	80.92

Table 1: Performance of different models. “Table” refers to the performance on the table part, and “Non-table” the non-table part; “Single” refers to the single model on the entire article; “Combined” refers to the combination of the two models on table and non-table parts; and “+Aux” refers to the use of the auxiliary model to eliminate misdetected toponyms.

For example, in the phrase *The Royal Melbourne Hospital*, the word *Melbourne* should not be detected as toponym according to the competition setting. This issue was also identified by Dredze et al. (2009).

In this paper, we use two methods to tackle this. The first is to train a single NER to detect toponyms and organization names together. The second is to train an organization name recognizer to correct misdetected toponyms in organization names.

We used the WikiNER (Nothman et al., 2012) dataset to train an organization detection model, and applied it to our dataset. Then we build an organization type set containing *Institute, School, Hospital* etc.. Finally, we re-label toponyms that are part of a corresponding organization name as non-toponyms.

3 Toponym Disambiguation

We used a support vector machine to disambiguate toponyms. For each detected toponym, we first search for it on Geonames, and keep the top 20 records as candidate results. Features are constructed from this, as follows:

- **History Result**: If the toponym appears in the training set, history result refers to the ranking of the number of times the Geonames ID appears as a standard answer. For instance, the toponym *Melbourne* appears 13 times in training, of which 12 occurrence have Geonames ID 2158177 and 1 has ID 7839805, so the history result feature for 2158177 is 1, 7839805 is 2, and all other Geonames IDs are 3.
- **Population**: The ranking of the population of the candidate.
- **GeoNames Feature Codes**: The feature class codes of Geonames records, e.g. *A* represents country, state, region,...; *P* city, village,...; etc.
- **Name Similarity**: The ranking of the string similarity of the toponym and Name item in each record.
- **AncestorsNames Correlation**: The ranking of matching words of the AncestorsNames item in each record.

4 Experiments and Results

4.1 Experiment Setting

The model architecture used for the toponym detection task is depicted in Figure 2. We use the Adam (Kingma and Ba, 2015) optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-9}$ and an initial learning rate of $1e^{-3}$. A dropout (Srivastava et al., 2014) rate of 0.5 is used to prevent overfitting. The hidden size (d) of the model is 300, and cross-entropy loss is used for training.

To compare different word embeddings, we use pre-trained 300-dimensional GloVe embeddings and pre-trained 1024 dimensional EMLo embeddings, respectively. We do not update the word embeddings during training.

Embedding	Precision	Recall	F1
GloVe	80.47	75.03	77.65
ELMo	**84.69**	**77.48**	**80.92**

Table 2: Performance of different word representations

4.2 Results

We randomly selected 10 articles from the training set to manually evaluate the table extraction method: 26 out of 27 tables were detected, and 3 non-table parts were misidentified as tables. 62% of tables are exactly accurate, or in other words, 38% tables have some lines misidentified.

Table 1 shows the subtask 1 performance (precision, recall and F1 score) of different models on the table and non-table parts. Strict and overlapping micro measures results are reported. In the strict measure, model outputs are considered to match with the gold standard annotations if they cover the exact same span of text; whereas in the overlapping measure, the model output is considered to match if it overlap in span with the gold-standard. From Table 1, we see that self-attention improves the results on both table and non-table parts, but is particularly effective for the table part. Experimental results on the non-table part are further improved by incorporating the auxiliary model. Finally, the combined model perform is better than a single model on entire articles.

Table 2 shows the subtask 1 performance of different word representations. From that, we find that using ELMo representation is much better than using GloVe embeddings. The reason is that our tokenization method separates many words like *I'm*, *let's*, which ELMo can generate a contextualized representation for, while GloVe cannot. Furthermore, there are many numbers and OOVs in the tables, the GloVe embedding for which is a random 300-dimensional vector that does not provide useful context information.

5 Conclusions

In this work, we presented a method for toponym detection and disambiguation in scientific papers, in the context of Sem-Eval 2019 Task 12, using an LSTM model and SVM model respectively. We extract tables from plain text, and train a dedicated model for each to improve overall performance due to the different structures of tables and the body of text. We also demonstrated the performance of the different models for toponym detection, with our final submission coming in 2nd (among 8).

References

Mariam Daoud and Jimmy Xiangji Huang. 2013. Mining query-driven contexts for geographic and temporal search. *International Journal of Geographical Information Science*, 27(8):1530–1549.

Mark Dredze, Partha Pratim Talukdar, and Koby Crammer. 2009. Sequence learning from data with multiple labels. In *Proceedings of the ECML/PKDD 2009 Workshop on Learning from Multi-Label Data*.

Claire Grover, Richard Tobin, Kate Byrne, Matthew Woollard, James Reid, Stuart Dunn, and Julian Ball. 2010. Use of the Edinburgh geoparser for georeferencing digitized historical collections. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1925):3875–3889.

Bo Han, Paul Cook, and Timothy Baldwin. 2014. Text-based Twitter user geolocation prediction. *Journal of Artificial Intelligence Research*, 49:451–500.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*.

Jochen L Leidner. 2008. *Toponym resolution in text: Annotation, evaluation and applications of spatial grounding of place names*. Universal-Publishers.

Arjun Magge, Davy Weissenbacher, Abeed Sarker, Matthew Scotch, and Graciela Gonzalez-Hernandez. 2018. Deep neural networks and distant supervision for geographic location mention extraction. *Bioinformatics*, pages i565–i573.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R. Curran. 2012. Learning multilingual named entity recognition from Wikipedia. *Artificial Intelligence*, 194:151–175.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2227–2237.

Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. 2015. Twitter user geolocation using a unified text and network prediction model. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 630–636.

Stephen Roller, Michael Speriosu, Sarat Rallapalli, Benjamin Wing, and Jason Baldridge. 2012. Supervised text-based geolocation using language models on an adaptive grid. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1500–1510.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Maria Vasardani, Stephan Winter, and Kai-Florian Richter. 2013. Locating place names from place descriptions. *International Journal of Geographical Information Science*, 27(12):2509–2532.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Annual Conference on Neural Information Processing Systems (NIPS 2017)*, pages 6000–6010.

Davy Weissenbacher, Arjun Magge, Karen O'Connor, Matthew Scotch, and Graciela Gonzalez. 2019. Semeval-2019 task 12: Toponym resolution in scientific papers. In *Proceedings of the 13th International Workshop on Semantic Evaluation*.

University of Arizona at SemEval-2019 Task 12: Deep-Affix Named Entity Recognition of Geolocation Entities

Vikas Yadav, Egoitz Laparra, Ti-Tai Wang, Mihai Surdeanu, Steven Bethard
University of Arizona
{vikasy,laparra,twang03,msurdeanu,bethard}@email.arizona.edu

Abstract

We present the Named Entity Recognition (NER) and disambiguation model used by the University of Arizona team (UArizona) for SemEval 2019 task 12. We achieved fourth place on tasks 1 and 3. We implemented a deep-affix based LSTM-CRF NER model for task 1, which utilizes only character, word, prefix and suffix information for the identification of geolocation entities. Despite using just the training data provided by task organizers and not using any lexicon features, we achieved 78.85% strict micro F-score on task 1. We used the unsupervised population heuristics for task 3 and achieved 52.99% strict micro-F1 score in this task.

1 Introduction

Geoparsing is the task of detecting geolocation phrases in unstructured text and normalizing them to a unique identifier, e.g. GeoNames[1] IDs. Although many automatic resolvers have been released in the past years, their performance fluctuates when applied to different domains (Gritta et al., 2018b). Most have also not been applied to and evaluated on scientific publications. The SemEval 2019 Shared Task 12: Toponym Resolution in Scientific Papers (Weissenbacher et al., 2019) aims to boost the research on geoparsing for the scientific domain by focusing on epidemiology journal articles.

The task includes three sub-tasks: toponym detection, toponym disambiguation, and end-to-end toponym resolution. The first one requires participants to detect the text boundaries of all toponym mentions in articles. In toponym disambiguation, the toponym mentions are known, and the resolver has to align them to their precise coordinates through GeoNames IDs. For the last sub-task, the resolver must perform both detection and disambiguation.

In this paper, we present the description of our system for SemEval 2019 Shared Task 12, in which we focus mainly on toponym detection. For this sub-task, we propose a recurrent neural network that combines word, character and affix information. By making use of the baseline provided by the organizers for toponym disambiguation, we also obtain results for the end-to-end sub-task.

2 Related Work

Toponym detection and resolution has been widely studied, and various systems (Gritta et al., 2018b) have been proposed for these tasks. Toponym detection has been implemented on texts from various sources like social media (Karagoz et al., 2016), PubMed articles (Magge et al., 2018) etc. Various named entity recognition (NER) systems including rule-based (Gritta et al., 2018b), machine learning-based (Karagoz et al., 2016), and deep learning-based (Magge et al., 2018) have been implemented for detecting toponyms.

The disambiguation step has been tackled previously using both supervised models and unsupervised heuristic based approaches. For example, Turton (2008) presented a rule based system for disambiguating locations from PubMed abstracts. Weissenbacher et al. (2015) presented results from *Population* and *Distance* heuristics (discussed in Section 4.3) for the disambiguation task on PubMed articles. The authors also presented an SVM model with population, distance and set of meta-data as input which achieved higher performance than both the individual heuristics. Gritta et al. (2018a) used a feedforward neural network approach for the disambiguation of geolocations.

[1] http://www.geonames.org/

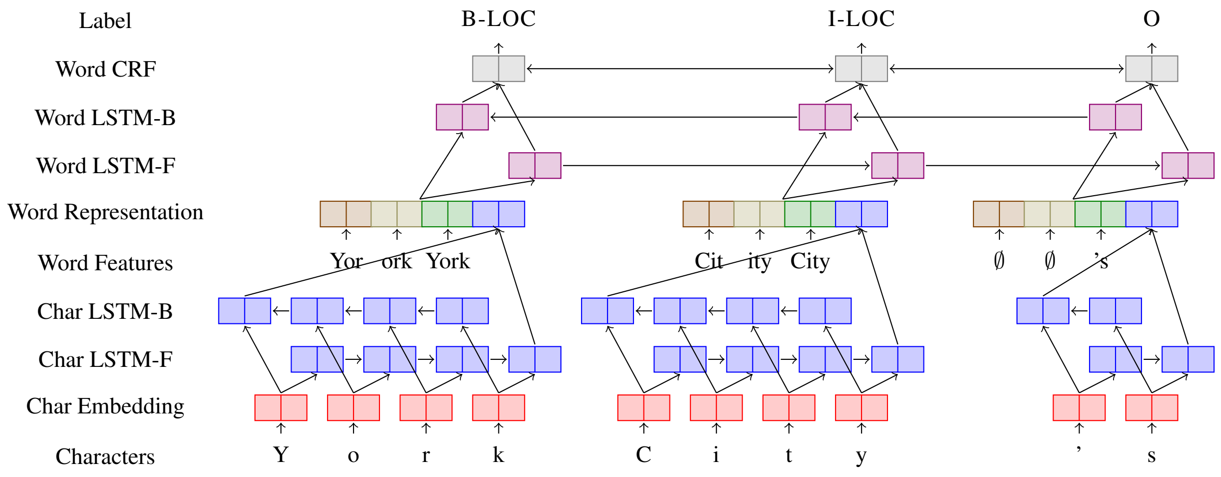


Figure 1: Word+character+affix neural network architecture from Yadav et al. (2018).

3 Data and Baseline

The corpus of the task is composed of 150 journal articles downloaded from PubMed Central. After removing the author names, acknowledgments and references, titles and body text were fully annotated. The annotators identified and labelled toponyms with their corresponding coordinates according to GeoNames. For cases not found in GeoNames, they used Google Maps and Wikipedia. If the coordinates of a toponym were not available in any of these resources the special value N/A was used. The data is provided in Brat format (Stenetorp et al., 2012). The organizers also released a strong baseline that combines the model by Magge et al. (2018) for toponomy detection and the *Population* heuristic described in (Weissenbacher et al., 2015) for disambiguation.[2]

4 Approach

4.1 Preprocessing

We used the tokenizer included in the baseline provided by the organizers as we observed it provided the best final results among other options (see Section 5.3). Again using baseline system preprocessing codes, we converted the data into CoNLL 2003 format (Tjong Kim Sang and De Meulder, 2003) for task 1. Following our prior work (Yadav and Bethard, 2018), we have used a BIO encoding instead of the IO encoding provided by the baseline system.

[2] https://github.com/amagge/semeval-ffnn-baseline

4.2 Toponym Detection

We used the model proposed by Yadav et al. (2018) for Named Entity Recognition (NER), shown in figure 1, which uses character, word and affix information. In this architecture, a word is represented by concatenating its word embedding, an LSTM representation over the characters of the word, and learned embeddings for prefixes and suffixes of the word[3]. Then another LSTM is used at the sentence level to give a contextual representation of each word. These representations of words in the sentence are given to a CRF layer to finally predict the NER label.

4.3 Toponym Resolution

Weissenbacher et al. (2015) presented two heuristics for disambiguation of geolocation - *Population* and *Distance*. These two heuristics are often used as features with other meta-data such as the user location meta-data in a Twitter account (Zhang and Gelernter, 2014), GenBank meta-data (Weissenbacher et al., 2015), etc.

In the *Population* heuristic, the system simply assigns the geonameID of the most populous[4] candidate for the current location. For the *Distance* heuristic, the system selects the candidate which is at the minimum distance from all candidates of all other toponyms in the same document. Many previous works (Weissenbacher et al., 2015; Zhang and Gelernter, 2014; Weissenbacher et al., 2019) have shown that the most populous location is often referenced more in the text documents and performs

[3] The affix vocabulary consisted of all three-character affixes that occurred at least 50 times in the training data.

[4] Population retrieved from the GeoNames database.

better than the distance heuristics. Thus, we use the *Population* heuristic as our disambiguation model.

5 Experiments

Using the original fully annotated training set, we achieved 77.3% strict micro-Fscore (mean performance of 3 runs) on the validation set. However, the organizers provided two additional large (but weakly) annotated NER datasets: *POS*, which contains sentences having at least 1 location phrase, and *NEG*, which has sentences with no mention of location entities. We experimented with both these datasets in both joint and transfer learning.

5.1 Joint Learning

In the joint learning experiment, we trained the model on a training set by concatenating the *POS* data with the original training data. In this configuration, we achieved 81.4% strict micro-F score (mean performance of 3 runs) on the validation set, a 4 point improvement over the original experiment.

5.2 Transfer Learning

In this experiment, we first trained our model on just the *POS* set and further fine tuned it on the original training data provided for the task. The intuition here was to use the weakly annotated data only to get a good initialization for the "real" training on the manually annotated data, rather than training on both together and possibly getting misled by the noise in the weakly annotated data. We achieved 83.7% strict micro-F score (mean performance of 3 runs) on the validation set. This is an improvement of 2.3 F over the simple joint learning experiment, and 6.4 F over the model using only the original training data.

5.3 Effects of Tokenization

The effect of tokenization on NER performance has been shown in the past (Akkasi et al., 2016; Xu et al., 2018). For this reason, we evaluated our model trained on the original training data, using various custom tokenization functions, and saw the strict micro-F1 score vary from 72% to 77% in the validation set.

The NLTK regexp tokenizer resulted in 70% strict F1-score. We wrote several rules to improve this tokenizer which further improved the performance by 4%.

Parameter	value
Word embedding (GloVe) size	300
Character embedding size	50
Affix embedding size	30
Word LSTM hidden state size	50
Character LSTM hidden state size	25
Learning rate	0.15
Learning rate decay	0.99
Batch size	100
Optimizer	SGD

Table 1: Hyperparameters for training the model.

However, the custom tokenization implemented by the shared task organizers in the baseline model performed the best, achieving 77% on the validation set when trained on just the original training data. In this case, we also wrote a few additional rules to improve the tokenization but achieved marginal improvements in the overall performance.

5.4 Hyperparameters

We trained the Yadav et al. (2018) model using the parameters in Table 1. For transfer learning from *POS* data, we first trained the model for 40 epochs. We then retrained this model on the original training data for 80 epochs with 20 as the early stopping patience. After training on the original training data, we retrained this model on train+development data for another 40 epochs. For the final evaluation, we submitted the models at epoch = 25, 35 and 40. Epoch 35 achieved the best performance among the three submissions.

The software is available at `https://github.com/vikas95/Pref_Suff_Span_NN`.

6 Results

We achieved the 4^{th} position in both task 1 (toponym detection) and task 3 (end-to-end toponym resolution) as shown in tables table 2 and table 3, respectively. Although it has been shown previously that adding lexicon features improves the overall performance of several NER models (Yadav and Bethard, 2018; Gritta et al., 2018b), we have focused on extraction of context information using LSTMs over character, word and affixes of the word. Hence, our resource-independent NER model achieves competitive results, despite not using any dictionary information. Also, we have just used the training data provided by the task organizers and did not use any external training data or

Team	strict Micro F	strict Macro F
DM_NLP	89.13	91.61
QWERTY	83.33	87.10
Newbee	80.92	87.11
Our model	78.75	84.52
THU_NGN	74.96	83.23
UNH	73.12	81.93
RGCL-WLV	49.13	61.96
NLP_IECAS	64.85	74.82

Table 2: Results of subtask 1 – toponym detection. We include the best Micro F-score and best Macro F-score of each team from their final 3 runs. Our model is ranked fourth, despited the fact that it uses no external knowledge.

Team	strict Micro F	strict Macro F
DM_NLP	0.7291	0.7749
QWERTY	0.7128	0.7551
Newbee	0.6545	0.7355
Our model	0.5299	0.6487
THU_NGN	0.5156	0.6131
NLP_IECAS	0.5223	0.6019

Table 3: Results of subtask 3 - end-to-end toponym resolution. Our system is again ranked fourth.

lexicon resources.

We used the unsupervised *Population* heuristic which is fast and simple to implement for disambiguating toponyms. As shown by Weissenbacher et al. (2015), feeding features like population, distance, and other meta-data to machine learning models often achieved higher performances. However, as shown here, the *Population* heuristic serves as a strong baseline for this disambiguation task.

7 Future Work

We plan to include the following features in our current model:

- Part of Speech (POS) features – as per the annotations guidelines, locations that were used as adjectives were not labelled in the annotation process. We will explore the effect of adding POS feature representation to the word, character and affix representations.

- Inclusion of geoname dictionary – our current approach is resource independent. We will include dictionary features in the next version of our model, to understand how much signal can be inferred from local information, and how much must come from world knowledge.

- Using domain-specific embeddings – we relied on pretrained GloVe embeddings for our submissions. In future versions of our software, we will explore domain-specific embeddings, i.e., trained on scientific texts, as well as contextualized embeddings such as FLAIR (Akbik et al., 2018).

8 Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency (DARPA) under the World Modelers program, grant number W911NF1810014. Mihai Surdeanu declares a financial interest in lum.ai. This interest has been properly disclosed to the University of Arizona Institutional Review Committee and is managed in accordance with its conflict of interest policies.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.

Abbas Akkasi, Ekrem Varoğlu, and Nazife Dimililer. 2016. Chemtok: a new rule based tokenizer for chemical named entity recognition. *BioMed research international*, 2016.

Milan Gritta, Mohammad Taher Pilehvar, and Nigel Collier. 2018a. Which melbourne? augmenting geocoding with maps. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1285–1296.

Milan Gritta, Mohammad Taher Pilehvar, Nut Limsopatham, and Nigel Collier. 2018b. Whats missing in geographical parsing? *Language Resources and Evaluation*, 52(2):603–623.

Pinar Karagoz, Halit Oguztuzun, Ruket Cakici, Ozer Ozdikis, Kezban Dilek Onal, and Meryem Sagcan. 2016. Extracting location information from crowdsourced social network data. *European Handbook of Crowdsourced Geographic Information*, 195.

Arjun Magge, Davy Weissenbacher, Abeed Sarker, Matthew Scotch, and Graciela Gonzalez-Hernandez. 2018. Deep neural networks and distant supervision for geographic location mention extraction. *Bioinformatics*, 34(13):i565–i573.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: A web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 102–107, Stroudsburg, PA, USA. Association for Computational Linguistics.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Ian Turton. 2008. A system for the automatic comparison of machine and human geocoded documents. In *Proceedings of the 5th Workshop on Geographic Information Retrieval*, pages 23–24. ACM.

Davy Weissenbacher, Arjun Magge, Karen O'Connor, Matthew Scotch, and Graciela Gonzalez. 2019. Semeval-2019 task 12: Toponym resolution in scientific papers. In *Proceedings of The 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Davy Weissenbacher, Tasnia Tahsin, Rachel Beard, Mari Figaro, Robert Rivera, Matthew Scotch, and Graciela Gonzalez. 2015. Knowledge-driven geospatial location resolution for phylogeographic models of virus migration. *Bioinformatics*, 31(12):i348–i356.

Dongfang Xu, Vikas Yadav, and Steven Bethard. 2018. Uarizona at the made1. 0 nlp challenge. *Proceedings of machine learning research*, 90:57.

Vikas Yadav and Steven Bethard. 2018. A survey on recent advances in named entity recognition from deep learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145–2158.

Vikas Yadav, Rebecca Sharp, and Steven Bethard. 2018. Deep affix features improve neural named entity recognizers. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 167–172.

Wei Zhang and Judith Gelernter. 2014. Geocoding location expressions in twitter messages: A preference learning method. *Journal of Spatial Information Science*, 2014(9):37–70.

Index